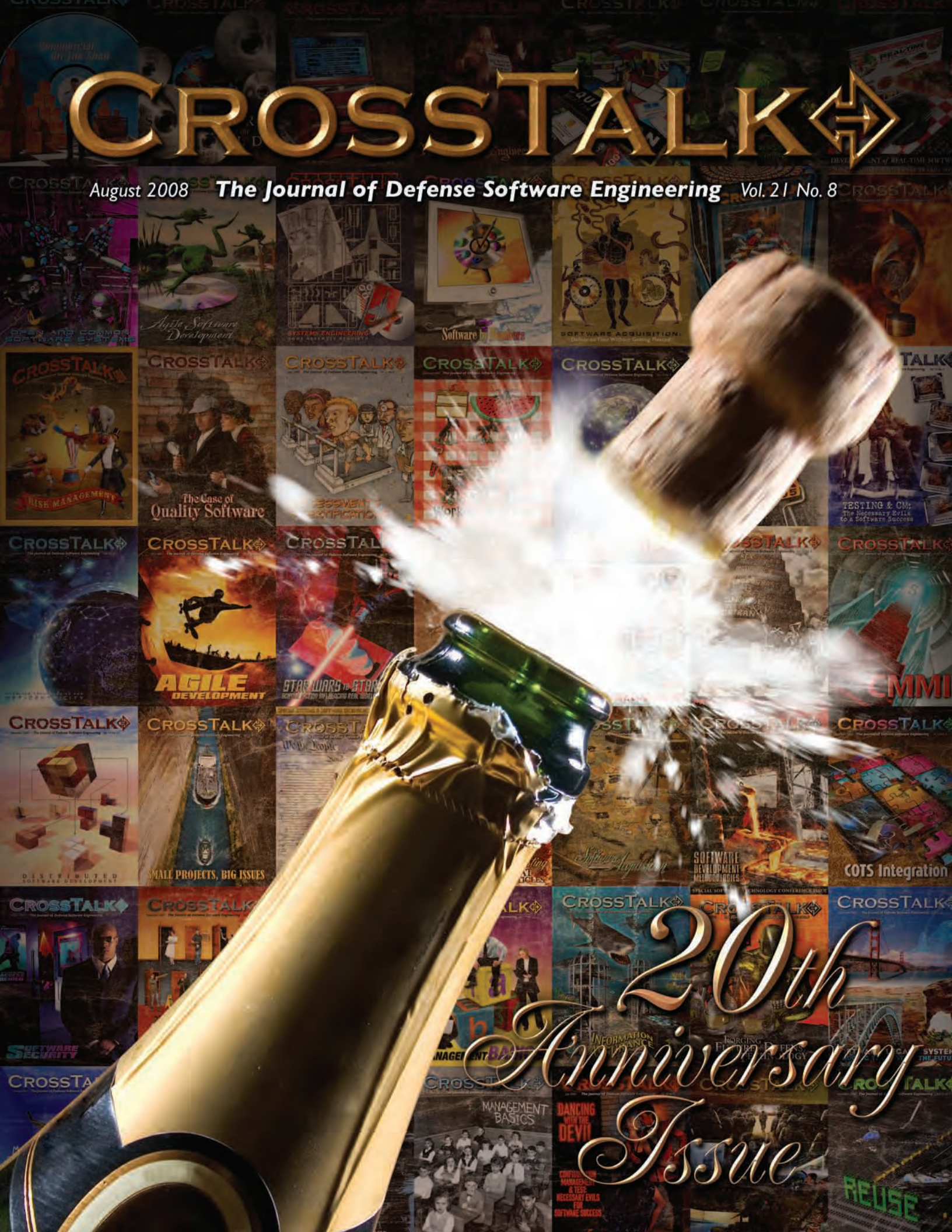
CROSSTALK
August 2008 The Journal of Defense Software Engineering Vol. 21 No. 8
20th Anniversary Issue

20th Anniversary

Software Engineering Technology

Open Forum

On The Cover

Cover Design by Kent Bingham

Additional art services provided by Janna Jensen

Departments

CrossTalk

Co-Sponsors:

DoD-CIO	*The Honorable John Grimes*
OSD (AT&L)	*Kristen Baldwin*
NAVAIR	*Jeff Schwalb*
76 SMXG	*Phil Perkins*
309 SMXG	*Karl Rogers*
DHS	*Joe Jarzombek*

Staff:

Managing Director	*Brent Baxter*
Publisher	*Kasey Thompson*
Managing Editor	*Drew Brown*
Associate Editor	*Chelene Fortier-Lozancich*
Article Coordinator	*Nicole Kentta*
Phone	(801) 775-5555
E-mail	stsc.customerservice@hill.af.mil
CrossTalk Online	www.stsc.hill.af.mil/crosstalk

CrossTalk, The Journal of Defense Software Engineering is co-sponsored by the Department of Defense Chief Information Office (DoD-CIO); the Office of the Secretary of Defense (OSD) Acquisition, Technology and Logistics (AT&L); U.S. Navy (USN); U.S. Air Force (USAF); and the U.S. Department of Homeland Security (DHS). DoD-CIO co-sponsor: Assistant Secretary of Defense (Networks and Information Integration). OSD (AT&L) co-sponsor: Software Engineering and System Assurance. USN co-sponsor: Naval Air Systems Command. USAF co-sponsors: Oklahoma City-Air Logistics Center (ALC) 76 Software Maintenance Group (SMXG); and Ogden-ALC 309 SMXG. DHS co-sponsor: National Cyber Security Division of the Office of Infrastructure Protection.

The USAF Software Technology Support Center (STSC) is the publisher of CrossTalk, providing both editorial oversight and technical review of the journal. CrossTalk's mission is to encourage the engineering development of software to improve the reliability, sustainability, and responsiveness of our warfighting capability.

S T S C

Subscriptions: Send correspondence concerning subscriptions and changes of address to the following address. You may e-mail us or use the form on p. 29.

517 SMXS/MXDEA
6022 Fir AVE
BLDG 1238
Hill AFB, UT 84056-5820

Article Submissions: We welcome articles of interest to the defense software community. Articles must be approved by the CrossTalk editorial board prior to publication. Please follow the Author Guidelines, available at <www.stsc.hill.af.mil/crosstalk/xtlkguid.pdf>. CrossTalk does not pay for submissions. Articles published in CrossTalk remain the property of the authors and may be submitted to other publications.

Reprints: Permission to reprint or post articles must be requested from the author or the copyright holder and coordinated with CrossTalk.

Trademarks and Endorsements: This Department of Defense (DoD) journal is an authorized publication for members of the DoD. Contents of CrossTalk are not necessarily the official views of, or endorsed by, the U.S. government, the DoD, the co-sponsors, or the STSC. All product names referenced in this issue are trademarks of their companies.

CrossTalk Online Services: See <www.stsc.hill.af.mil/crosstalk>, call (801) 777-0857 or e-mail <stsc.webmaster@hill.af.mil>.

Back Issues Available: Please phone or e-mail us to see if back issues are available free of charge.

Twenty Years of CrossTalk

It is my privilege to congratulate CrossTalk on 20 years of publishing articles for and by the defense software community. Longevity in any field is not easy, but longevity in the fast-paced world of software is indeed a notable accomplishment. The occurring changes during the past 20 years are far too countless to mention, yet they have all contributed to the present state of defense software today. Likewise, those of us currently involved with CrossTalk stand on the shoulders of the authors, editors, sponsors, and publishers who went before us, all of whom significantly contributed in assisting CrossTalk in fulfilling its mission of informing readers of new industry trends, proven methodologies, cutting-edge technologies, innovative practices, and, of course, lessons learned.

In my role as Director of 309 Software Maintenance Group, I feel a renewed resolve and duty to see CrossTalk thrive in its role to disseminate information to a community with such diverse and crucial responsibilities. CrossTalk has proven its worth by educating countless engineers and adding value to our processes and projects. The continued publication of CrossTalk can only further enhance those advances.

In this special 20th anniversary issue, Gerald M. Weinberg takes a bi-directional view of where defense software had been and where it is going in his article *What Have 20 Years Accomplished and What Is Left to Accomplish in the Next 20 Years?*, while Dr. Linda Ibrahim provides insightful thoughts about the progress of the industry as a whole in *A Process Improvement Commentary*.

Watts S. Humphrey cautions readers in his article, *The Process Revolution*, not to repeat mistakes and to build reliability and consistency into your processes by consistently recording and learning from history. In the same vein, Paul Kimmerly's article, *Heroes: Carrying a Double-Edged Sword*, discusses the benefits and limitations of heroes in process improvement efforts.

You may have heard the saying *the more things change the more they stay the same*. Interestingly, Dr. Alistair Cockburn explores the merits of this platitude in a software sense by reviewing Agile practices in his article *Good Old Advice*, and focuses on Agile's applicability in today's world.

Conversely, Jamie Hohman and Dr. Hossein Saiedian provide guidance in their article, *Wiki Customization to Resolve Management Issues in Distributed Software Projects*. This article adds new dimensions to the dynamics of traditional project management and provides methods of easing the stresses brought on by trying to manage distributed projects.

Lastly, don't miss Gary A. Petersen's witty walk down CrossTalk's memory lane in his article, CrossTalk: *The Long and Winding Road*, as he examines the maturity and growth of the magazine as it developed from a diminutively distributed black-and-white copy to its current form and shape.

It is my hope that CrossTalk continues to publish insightful articles for another 20 years and that together we can celebrate just as many accomplishments in the defense software community with the 40th anniversary issue.

Karl Rogers
Ogden-Air Logistics Center, Co-Sponsor

The Process Revolution

Watts S. Humphrey
The Software Engineering Institute

To mature software engineering, we must learn from history, and to learn from history, we must define what we do when we develop software and how we do it. When we produce a process specification and define measures for the key process parameters, we are beginning to build the basis for reliably and consistently recording and learning from history. That is why the process revolution is so important.

Judging by the history of other fields, software engineering and computer science are still in their infancy. We have yet to build an accepted foundation of proven and generally followed principles, and there is little in the way of defined and widely taught practices to guide working professionals. When new problems arise in the software field, the tendency is to invent some new language, tool, or method to solve them.

The true measure of maturity for our field will be when we stop dismissing the lessons of history and start building upon them. While there are many proven truths in the practice of software engineering, few software people accept them until they have personally suffered through painful and expensive failures. Unfortunately, as our field has grown and expanded, the number of ways to fail has become so large that learning through failure is no longer a viable way to advance the field.

To have a consistently effective software industrial capability, we must use defined and precise methods and well-understood procedures. Learning from history is the only way to develop such methods. While I was not the first or only person to start introducing process methods into industrial software engineering practice, I was fortunate enough to be involved in some of the early key events. This brief history provides insight into what happened, why it happened, and what is likely to happen next.

Software Factories

Since the beginning of the computer age, software projects have been troubled. Schedules were rarely met, costs were unpredictable, and total project failures were common. Of the many attempts to address this situation, one particularly ill-conceived idea was an attempt approximately 20 years ago to solve software's problems by establishing what were called *software factories*. The rationale for this idea, as I understood it, was that since typical manufacturing factories regularly produced products with predictable costs and schedules, the software community should emulate factory practices. The hope was that this would somehow enable them to achieve factory-like results.

While at IBM in the early '80s, I visited Japan and met with several software groups. One proudly took me on a tour of their software factory. It was comical. Before entering the factory's cleanroom, we had to put on smock coats just like the ones worn in semiconductor cleanrooms. Inside were dozens of software developers lined up at rows of identical work stations. While this did actually look like a factory, I had trouble believing that this factory charade could have anything but a negative impact on software development creativity and productivity. Sure enough, in a relatively short time, the idea of the software factory disappeared into the growing dustbin of software's simplistic and superficial methods that do not address fundamental problems.

Mike Fagan

A more thoughtful approach to software's problems had actually been initiated a few years earlier when Mike Fagan published his landmark paper on software inspections [1]. In it, he not only described a way to economically improve software quality, but he provided a coherent description of software process principles and explained how process concepts can be used to improve the performance of software groups. Fagan's ideas had already been tested on several projects and were found to provide truly extraordinary results. The benefits were so substantial, in fact, that he was given a $100,000 outstanding contribution award by IBM corporate management.

The 1986 Attitude

In my final years at IBM, my views were heavily influenced by Fagan and Al Pietrasanta [2]. I even drafted a short paper for the *IEEE Spectrum*, outlining a vision of how, by improving the software process, we could greatly improve the performance of software groups [3]. When I joined the Software Engineering Institute (SEI) in 1986, Fagan's inspection process was widely used at IBM but practically nowhere else. Furthermore, the ideas that he, Leon Osterweil, Bob Balzer, and others were discussing about defining and using processes were not understood by most of the software community or even recognized as important [4]. In fact, when I talked about software process at the time, the typical question I was asked was, "What in the world is a software process?"

This attitude posed a problem. Fortunately the SEI then had a shortage of experienced business managers so I was asked to run the finance and administrative operations. I agreed to do so on two conditions: 1) that the assignment would be temporary, and 2) that I could start a process group of my own. That is how we started the SEI Process Program.

The Process Maturity Model

After joining the SEI, I was asked to work with Bill Sweet of the SEI and R.K. Edwards, G.R. LaCroix, M.F. Owens, and K.P. Schultz of the Mitre Corporation to define a way to identify the U.S. Air Force's most competent software vendors. One of the key background documents we were given was an Air Force study of 17 major projects[1]. We found that the average schedule overrun for these projects was 75 percent, meaning that a typical four-year project would be completed in seven years. Furthermore, not one of the projects was completed on time or within the original cost target. We soon realized that we did not have a selection problem – all of the vendors were failing – so the fundamental problem was capability. How could these organizations improve? How could we measure their improvement? How could we motivate

continuous improvement?

We then considered the causes of project failure as well as the characteristics typically associated with the infrequent successes. Even though all of the systems that we had studied were complex and had a myriad of technical issues, we found that the principal causes of project failure were not technical. Failing groups typically had no plans, were not tracking or managing the plans that they had, or had other very basic project management failings. While we all agreed that technical issues were important, we could not identify any language, support environment, or design method that would consistently distinguish successful projects from failed projects. The only exception was the use of Fagan inspections. This practice was closely associated with successful projects and was not found in failed projects.

While we realized that the processes used by software organizations were key, we also found that the processes then discussed in the software literature were entirely technical and that none of them addressed the topics we had found most important to project success. Once we realized that the process characteristics that would distinguish between successful and failed software projects concerned management practices, we listed the key practices in question form and tried to construct a way for acquisition groups to evaluate organizations based on a questionnaire [5]. Our questionnaire had 85 questions and our contention was that a high success rate on the questions would correlate to a high likelihood of project success.

Since a typical acquisition might involve 10 or more vendors, we then had to provide a way to rank these organizations according to 850 questionnaire answers. This required some kind of ranking system. We soon developed the idea of using a maturity model such as the one devised by Philip B. Crosby [6]. We then produced a technical report for the Air Force to complete our assigned task and to provide an initial assessment method. These early ideas were published in a technical paper and a book [7].

The Capability Models

What was most surprising about the maturity model was the high level of interest by the software development community, particularly among Department of Defense (DoD) contractors. Clearly, there was a widespread need for an easily understandable way to evaluate organizational capability, to identify improvement priorities, and to measure improvement progress. We did, however, need two things before process maturity concepts would be widely accepted and used.

The first need was to have the evaluation method accepted by the key users. We accomplished this by forming an advisory board for model development that would include key representatives from all stakeholder communities. This board had to be composed of knowledgeable representatives from industry and government, and the SEI had to be willing to defer to their views. The advisory board was formed and the resulting maturity framework was called the Capability Maturity Model® (CMM®) and subsequently the Capability Maturity Model Integration (CMMI®) [8].

The second key requirement was strong backing by the DoD. This was soon forthcoming in the form of a directive that all future DoD software contractors would have to demonstrate a CMM Level 3 capability or better to win contracts.

Process Integration

As the CMM became more widely used, other groups began adopting the idea. For example, maturity models were developed for systems engineering, acquisition, and people management. Soon, the number and type of maturity models became confusing, and talk began of establishing a common maturity-model framework or architecture.

At the same time, some people in the DoD recognized that the real issue was the maturity of the entire systems development process and that software, hardware, and systems engineering were all parts of a larger and more integrated framework. At that point, the DoD sponsored – and the SEI led – a major effort to develop an integrated, multi-disciplined model to replace the CMM. This model is called the CMMI.

Where We Are Now and What Is Next

While the CMMI-based process revolution has had a major and positive impact on the performance of software organizations, the story is not yet over. There are still many organizations that have not adopted the CMMI or addressed process improvement. We are also beginning to see some organizations that have a high CMMI rating that are not performing as well as expected. This situation suggests that there are three issues: support, implementation, and omission.

The support issue refers to the long time required for adoption of any new method or technology and the need for a continuing support and enhancement effort during this time. One example of this problem is illustrated by the Fagan inspection process at IBM. With Fagan's help, development managers were convinced that inspections would sharply cut test time, shorten development schedules, and improve product quality. Many inspection courses were given and the inspection process was implemented in all of the development laboratories.

Some years later, I found that several laboratories had stopped doing inspections. On investigation, we found that after inspections had been in place for several years and many of the managers and developers had changed jobs, people started to complain about the time that the inspections were taking. The inspection advocates explained that this method was important in holding down testing costs. However, since testing was not then seen as a problem, management decided to make inspections optional. Quality then rapidly deteriorated and testing times grew accordingly. With no defined process or measures, however, the history was lost and nobody recognized what had happened.

The implementation issue concerns the skill and competence of the groups using CMMI methods. This issue relates to the support issue and concerns the problems caused by addressing symptoms instead of causes. While the evidence for poor software performance might include lack of documented plans, ill-defined processes, or incomplete data, these are only symptoms. Fixing the symptoms might get a suitable maturity rating but it does not mean that the plans were properly made and used, the processes actually followed, or the data analyzed and used. The problem is that when the DoD started requiring CMMI maturity levels for contracts, some groups found that it was easier to produce artifacts than to change engineering behavior. However, with processes, the connection between symptom and performance is behavior, and unless behavior changes, performance will not change. Without an understanding of history, it is hard to change behavior.

The omission issue concerns additional process topics that should be addressed. In developing the initial process maturity concepts, we focused on project management issues because they were the most obvious source of project problems. We also assumed that management's motivation in using these methods would be to

® CMM, CMMI, and Capability Maturity Model are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

improve organizational performance. If, however, management was primarily interested in achieving a maturity level, it would not be as likely to detect the difference between merely addressing a symptom and actually improving performance.

To improve performance, we had to address the behavior of the managers and developers. We did not initially recognize that technical and team-level issues would be critically important. Once we did, however, we extended the CMMI concepts to the personal and team levels with the Personal Software Process℠ (PSP℠) and the Team Software Process℠ (TSP℠). These methods are designed to address the behavior issue at the developer and team levels [9]. We have also started to broaden the TSP to cover systems engineering, hardware development, and the other fields that are often involved in a systems-development project. In the video game business, for example, this includes artists and game designers as well as software developers. To be most effective, such integrated teams need an integrated process framework that can support a highly collaborative working style.

Conclusions

The process revolution is now in a situation much like that of a pond with a falling water level. While the most significant rocks and stumps may have been removed, new rocks will keep appearing. With improved management practices, the rocks we are now seeing concern quality management, integrated systems management, integrated development teams, and team member practices. Early work on these topics suggests that we now face another equally productive and exciting cycle in the process revolution. After these next issues are brought under control, the heretofore concealed-but-critical rocks in the process pond that concern safety, security, architecture, design, and domain expertise will almost certainly be revealed.

What is clear from this story is that learning from history works. While our field continues to advance and the future promises to be exciting, there are useful lessons to be learned from experience. Every project is a new experience, but not every part of every project is new. Where similar work has been done before, we need to learn from and build on that experience. To do this, however, we must define, plan, measure, track, and report on what we do. Only then will our experience be available in a form that is useful to us and to others. This is true for our organizations, for our projects, for our teams, and for us.◆

℠ Personal Software Process, Team Software Process, PSP, and TSP are service marks of Carnegie Mellon University.

Acknowledgements

Many people have contributed to the progress that has been made in the process revolution field and it is impossible to name them all. However, one of the people who had a profound influence on this field and who helped me personally was Al Pietrasanta. My first exposure to Pietrasanta's work was when he developed an early course to show programming managers how to plan and manage their work. This course was taught to more than 1,000 IBM managers, and had a major impact on IBM's ability to deliver the software required to support the 360 and 370 lines of computing systems in the 1960s and 1970s. Unfortunately, he died an untimely death a number of years ago.

The second person who has made major contributions to this field is Mike Fagan. He wrote the earliest paper that I am aware of on how to define and use processes to plan and manage development work. His early work profoundly changed the way IBM managed software quality and led to a new way to view software defects and defect management. We all owe Mike Fagan a debt of gratitude for his perceptive leadership.

For this article, I have been fortunate to have several well-informed and helpful reviewers. I particularly thank Dan Burton, Bob Cannon, Tim Chick, Jim McHale, Julia Mullaney, Jim Over, and Bill Peterson for their helpful comments and suggestions. I also much appreciate the helpful guidance of the CrossTalk staff.

References

1. Fagan, Michael E. "Design and Code Inspections to Reduce Errors in Program Development." IBM Systems Journal 15.3 (1976): 182-211.
2. Pietrasanta, Alfred M. On the Management of Computer Programming. Ed. George F. Weinwurm. Auerbach Publishers, 1970.
3. Humphrey, Watts S. "Speakout: We Can Program SDI." IEEE Spectrum Apr. 1986.
4. Osterweil, Lee J. Software Processes Are Software, Too. Proc. of the 9th International Conference on Software Engineering. Institute of Electrical and Electronics Engineers (IEEE) Computer Society Press. Apr. 1987.
5. Humphrey, W.S., and W.L. Sweet. "A Method for Assessing the Software Engineering Capability of Contractors." SEI Technical Report CMU/SEI-87-TR-23. Sept. 1987.
6. Crosby, Philip B. Quality Is Free: The Art of Making Quality Certain. New York: Mentor, New American Library, 1979.
7. Humphrey, Watts S. Managing the Software Process. Reading, MA: Addison Wesley, 1989.
8. Paulk, Mark C., Charles V. Weber, Bill Curtis, and Mary Beth Chrissis. The Capability Maturity Model. Reading, MA: Addison Wesley, 1995.
9. Humphrey, Watts S. Winning With Software: An Executive Strategy. Reading MA: Addison Wesley, 2002.

Note

1. Unfortunately, I never kept a copy of this report and I do not believe it was ever published. It might even have been classified.

Additional Reading

1. Humphrey, Watts S. "Characterizing the Software Process: A Maturity Framework." IEEE Software Mar. 1988: 73-79.
2. Chrissis, Mary Beth, Mike Konrad, and Sandy Shrum. CMMI, Second Edition. Reading MA: Addison Wesley, 2007.

About the Author

Watts S. Humphrey joined the SEI after his retirement from IBM in 1986. He established the SEI's Process Program and led development of the CMM for software, the PSP, and the TSP. During his 27 years with IBM, he managed all IBM's commercial software development and was Vice President of Technical Development. He holds graduate degrees in physics and business administration. He is an SEI Fellow, an Association of Computing Machinery member, an IEEE Fellow, and a past member of the Malcolm Baldrige National Quality Award Board of Examiners. In a recent White House ceremony, the President awarded him the National Medal of Technology.

SEI
4500 Fifth AVE
Pittsburgh, PA 15213-2612
Phone: (412) 268-6379
Fax: (412) 268-5758
E-mail: watts@sei.cmu.edu

Good Old Advice

Dr. Alistair Cockburn
Humans and Technology

From Winston Royce, Fred Brooks, Gerald Weinberg, and the speakers at the 1968 North Atlantic Treaty Organization (NATO) conference on software engineering to Barry Boehm, Victor Basili, Capers Jones, and Tom DeMarco, the best-known writers have been making the same recommendations, which we still ignore today.

"It's new, it's improved, it's old-fashioned," sings Tom Waits in "Step Right Up." How well he captures the evolution of software development!

With the benefit of hindsight, we can see that the best-known writers in the software field have been advocating the same four recommendations written in the agile manifesto for decades (see The Agile Manifesto section).

The older writers were ignored for decades while people searched for mechanical replacements for the key elements in developing software: thinking and communicating. But that's a separate story.

It Is About People

In the legendary "Psychology of Computer Programming," published in 1971, Gerald Weinberg starts his preface with these prescient words, "This book has only one major purpose – to trigger the beginning of a new field of study: computer programming as a human activity ... " [1].

His hope lay dormant for 15 years, until Tom DeMarco and Tom Lister wrote the equally legendary, "Peopleware" [2]; Raymonde Guindon and Bill Curtis wrote their landmark paper on cognitive strategies in design [3]; Turing Award winner Peter Naur wrote about programming as *theory building* [4]; and Fred Brooks wrote the oft-quoted but still not followed, "No Silver Bullet" [5].

People Trump Processes

Those authors established that software development is deeply intertwined with the mental and social processes of the people involved. As Weinberg put it, "I've learned a basic principle of the psychology of management: managers who pay attention to people get good results" [1].

Barry Boehm and Capers Jones captured details of projects over the years and gave us startling numbers. From Boehm's studies come the following [6]:

- Personnel/team capability were the most significant cost drivers, affecting productivity by 353 percent.
- Process maturity was 12th, with a 143 percent effect.

From Capers Jones' studies [7]:

- Upper management and staff experience contribute 120 percent to productivity.
- Effective methods/processes contribute only 35 percent.

Capers Jones also cites the following reverse effect:

- Staff inexperience: negative 177 percent.
- Ineffective methods/processes: negative 41 percent.

In other words, we have long known that success is governed more by who you hire, how they interact with each other, and how you treat them. Yet despite the data and the advice, we keep retreating to the weak hope that some new process, tool, or technique will relieve the need for talent and communication.

People Mean Communication

Once a certain amount of talent is in place, the speed of the project is directly related to the speed at which information moves between minds. Briefly, *anything that slows the movement of ideas between minds slows the project*.

That includes – most particularly – such issues as morale, trust, pride in work, personal safety, and the amount of face-to-face communication available.

Participants at the famous 1968 NATO conference on software engineering said as much. On page 53 of the conference report, J.N. Buxton says:

> We could use more and better and faster communication in a software group as a partial substitute for a science of software production. ... if I'm setting up a software group to carry out a project I'm extremely careful that all the people working on it are close personal friends, because then they will talk together frequently, and there will be strong lines of communication in all directions. [8]

T.J. Allen, of the Sloan School of Management, studied the question of communication in research and development labs [9]. He found that communication dropped steeply when people had to walk more than 10 meters to a colleague's desk – roughly the length of a school bus. Allen constructs what he considers an optimal research and development workspace. It is full of osmotic communication [10], dense conversation and chance overhearing, and strikingly resembles what we now refer to as a *war room*. Allen's results have been reproduced several times over the years by, among others, G. Olson and J. Olson [11].

Physical proximity not only increases the frequency but also the bandwidth of communication. J.C. McCarthy and A.E. Monk [12] detail how proximity provides people additional coordinated communication channels such as facial and hand gestures, vocal inflection, physical distance cues, cross-modality timing, and real-time question and answer.

Old Riddles Solved

This point is illustrated using the following two experiences from the 1960s. They were almost certainly puzzling when they occurred, but with our current understanding of software development, we can now see how these seemingly anomalous stories make good sense.

Experience 1: Gerald Weinberg writes the following:

> [At] a large university computing center ... a large common space was provided near the return window so that the students and other users could work on their programming problems. In the adjoining room, the center provided a consulting service for difficult problems, staffed by two graduate assistants. At one end of the common room was a collection of vending machines ... the noise from the revelers congregating at the machines often became more than some of the workers could bear ... [The computing center manager] went to investigate their complaint ... Without more than 15 seconds of observation, he went back to his office and inaugurated action to have the machines removed to some remote spot. The week after the machines had been removed – and signs urging quiet had been posted all around – the manager

> received another delegation ... They had come to complain about the lack of consulting service; and, indeed, when he went to look for himself, he saw two long lines extending out of the consulting room into the common room. He spoke to the consultants to ask them why they were suddenly so slow in servicing their clients ... For some reason, they said, there were just a lot more people needing advice than there used to be ... After some time, he discovered the source of the problem. It was the vending machines! When the vending machines had been in the common room, a large crowd always hovered around them ... they were chatting about their programs. ... Since most of the student problems were similar, the chances were very high that they could find someone who knew what was wrong with their programs right there at the vending machines. Through this informal organization, the formal consulting mechanism was shunted, and its load was reduced to a level it could reasonably handle. [1]

It is to allow for just such serendipitous exchanges that many companies now put coffee machines, whiteboards, and chairs at various places in the buildings.

Experience 2: In the following example, Dee Hock presents us with a much more puzzling scenario with the development of the first Visa credit card clearance system in the 1960s:

> We rented cheap space in a suburban building and dispensed with leasehold improvements in favor of medical curtains on rolling frames for the limited spatial separation required ... Swiftly, self-organization emerged. An entire wall became a pinboard with every remaining day calendared across the top. Someone grabbed an unwashed coffee cup and suspended it on a long piece of string pinned to the current date. Every element of work to be done was listed on a scrap of paper with the required completion date and name of the person who had accepted the work. Anyone could revise the elements, adding tasks or revising dates, provided that they coordinated with others affected. Everyone, at any time, could see the picture emerge and evolve. They could see how the whole depended on their work and how their work was connected to every other part of the effort. Groups constantly assembled in front of the board as need and inclination arose, discussing and deciding in continuous flow and then dissolving as needs were met. As each task was completed, its scrap of paper would be removed. Each day, the cup and string moved inexorably ahead. Every day, every scrap of paper that fell behind the grimy string would find an eager group of volunteers to undertake the work required to remove it. To be able to get one's own work done and help another became a sought-after privilege ... Leaders spontaneously emerged and re-emerged, none in control, but all in order. Ingenuity exploded. Individuality and diversity flourished. People astonished themselves at what they could accomplish and were amazed at the suppressed talents that emerged in others. Position became meaningless. Power over others became meaningless ... a community based on purpose, principle, and people arose. Individuality, self-worth, ingenuity, and creativity flourished and as they did, so did the sense of belonging to something larger than self, something beyond immediate gain and monetary gratification. No one ever forgot the joy of bringing to work the wholeness of mind, body, and spirit; discovering in the process that such wholeness is impossible without inseparable connection with the others in the larger purpose of community effort. No one ever replaced the dirty string and no one washed the cup ... The BASE-1 system came up on time, under budget, and exceeded all operating objectives. [13]

At the time, Hock's method must have seemed like madness. With the benefit of 40 more years of investigation, we might say that he had created an early version of Scrum [14].

All these experiences point to one thing: *Attend to the individuals and their interactions.*

It Is About Software

If building software is about the people doing the building, it is also about the software being built. Software has a merciless quality to it that is not present in any requirement or design document. Whether it runs or not depends on exactly what is written, not on how fervently the authors believe it will.

The best known writers in our field have been telling us for decades to *write some software* and then *learn from writing it.*

Winston Royce wrote the much (but falsely) maligned "Managing the Development of Large Software Systems" [15]. Rereading it in the light of modern agile development, we can see it as a model of clarity, not expressing at all what most people think it does.

Two commonly shown models of software development are ascribed to Royce: one with no feedback arrows at all between development stages, the other with feedback arrows from each stage to the previous. *These two pictures were never presented as even faintly possible actual development processes!* They appear only as illustrative stepping stones leading to a carefully drawn iterative model.

More interesting, though, is that after deriving the iterative model halfway through the article, Winston Royce says, "The remainder of this discussion presents five additional features that must be added to this basic approach to eliminate most of the development risks" [15].

Of these five features, fully two of them advise the manager to start programming *before* the analysis is done! The second piece of advice is:

> Attempt to do the job twice – the first result provides an early simulation of the final product. Without this simulation the project manager is at the mercy of human judgment. With the simulation, he can at least perform experimental tests of some key hypotheses and scope down what remains for human judgment. [15]

The 1968 NATO conference participants said very much the same thing. A.G. Fraser reported the following:

> Design and implementation proceeded in a number of stages ... Each stage produced a useable product and the period between the end of one stage and the start of the next provided the operational experience upon which the next design was based ... The first stage did not terminate with a useable object program but the process of implementation yielded the information that a major design change would result in a superior and less expensive final product. During the

> second stage the entire system was reconstructed; an act that was fully justified by subsequent experience ... The final major design change arose out of observing the slow but steady escalation of complexity in one area of the system. [8]

The participant J. Smith put it more sarcastically:

> I'm still bemused by the way they attempt to build software ... All documents associated with software are classified as engineering drawings. They begin with planning specification, go through functional specifications, implementation specifications, etc., etc. This activity is represented by a PERT chart with many nodes. If you look down the PERT chart you discover that all the nodes on it up until the last one produces nothing but paper. It is unfortunately true that, in my organization, people confuse the menu with the meal. [8]

Victor Basili, with A. Turner, formally proposed an evolutionary coding strategy in their 1975 paper conspicuously titled, "Iterative Enhancement: A Practical Technique for Software Development" (their skeletal subproject is referred to these days as a *walking skeleton* [16]):

> This paper recommends the iterative enhancement technique as a practical means of using a top-down, stepwise refinement approach to software development ... This technique begins with a simple initial implementation of a properly chosen (skeletal) subproject which is followed by the gradual enhancement of successful implementations in order to build the full implementation. [17]

In his famous paper, "No Silver Bullet," Fred Brooks described that technique as *growing* the software:

> I have seen the most dramatic results since I began urging this technique on the project builders in my software engineering laboratory class. Nothing in the past decade has so radically changed my own practice, or its effectiveness ... I find that teams can *grow* much more complex entities in four months than they can *build* ... The morale effects are startling. [5]

Craig Larman has documented the pedigree of such incremental-iterative development extensively [18]. Our industry experts have been recommending it for 40 years – but most project teams still find it a hard sell to their management.

The short form of the recommendation is to *put the mercilessness of running software to good use – write, and learn from the writing.*

It Is About the Users

Starting with Winston Royce again, one of his five steps (including his capital letters) advises: "INVOLVE THE CUSTOMER. For some reason, what a software design is going to do is subject to wide interpretation, even after previous agreement" [15].

Fred Brooks writes:

> I would go a step further and assert that it is really impossible for clients, even working with software engineers, to specify completely, precisely, and correctly the exact requirements of a modern software product before having built and tried some versions of the product they are specifying. [5]

In other words, it does not matter what they told you they want, when they finally see it, they will understand their world differently and the software must change to match that new understanding.

The researchers M. Keil and E. Carmel studied the effects of links to the users. They asked managers who had been on two projects, one successful and one less successful, to compare the forms of access to users. They wrote the following:

> Results of the analysis revealed that in 11 of the 14 paired cases, the more successful project involved a greater number of links than the less successful project ... This difference was found to be statistically significant in a paired t-test ($p<0.01$). [19]

Paying attention to the users goes beyond merely getting it right. Users constitute a powerful lobbying force that can shift a project from irrelevant to top priority or shift the developers from being seen as an enemy force to an ally – to the benefit of both the project and the users.

In other words, *get inside the heads of your users and customers, and get them on your side.*

It Takes (Re)Planning

It would be nice if we could properly plan these projects in advance, but the most recognized writers in our field warn us against this, as Royce says in the following:

> ... [there are] five steps that I feel necessary to transform a risky development process into one that will provide the desired product. I would emphasize that each item costs some additional sum of money. If the relatively simpler process without the five complexities described here would work successfully, then of course the additional money is not well spent. In my experience, however, the simpler method has never worked on large software development efforts and the costs to recover far exceeded those required to finance the five-step process listed. [15]

At the 1968 NATO conference, E. David said:

> Computing has one property, unique, I think, that seriously aggravates the uncertainties associated with software efforts. In computing, the research, development, and production phases are often telescoped into one process. In the competitive rush to make available the latest techniques ... we strive to take great forward leaps across gulfs of unknown width and depth ... Thus, there are good reasons why software tasks that include novel concepts involve not only uncalculated but also uncalculable risks. [8]

D. Ross added:

> The most deadly thing in software is the concept, which almost universally seems to be followed, that you are going to specify what you are going to do, and then do it. And that is where most of our troubles come from. [8]

Fred Brooks puts it more simply: "Plan to throw one away; you will, anyhow" [5].

Even in civil and aeronautical engineering, the sage advice is to plan to replan. Dr. A. Laufer, editor-in-chief emeritus of NASA's Academy Sharing Knowledge e-zine, summarized it: "Plan and control to accommodate change ... Start planning early and employ an evolving planning and control process" [20].

The short summary of the last 40 years of experience is this: *Plan coarse-grained long-term and fine-grained short term, and find a way*

to re-plan quickly – because you'll have to.

One Process Will Not Fit All

People in our field search continually for the one, true process which, if we could only find it, would serve as the base for every project and also as master training material for newcomers to the field.

In [21], I investigated this question directly. I discovered that processes must change as technologies change, and also change to fit varied types of projects. How many types of projects are there? Capers Jones estimated there to be at least 37,400 different categories of projects [7]. No wonder no single methodology or process will fit them all. *Do not believe in any single process or methodology because each works only in a particular and limited context.*

The Agile Manifesto

In 2001, the authors of the Manifesto for Agile Software Development, made the same recommendations in their own way [22]. They wrote:

> We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
> - Individuals and interactions over processes and tools.
> - Working software over comprehensive documentation.
> - Customer collaboration over contract negotiation.
> - Responding to change over following a plan.
>
> That is, while there is value in the items on the right, we value the items on the left more. [22]

The 12th and final principle in the manifesto says:

> At regular intervals, the team reflects on how to become more effective, then tunes and adjusts their behavior accordingly. [22]

Summary

I derived the four values of the Agile Manifesto from much older, recognized, and highly reliable articles. The point I wish to make is that the authors of the Agile Manifesto *did not* throw out all previous experience as they wrote it. On the contrary, what they did was cherry-pick four of the most important issues from among hundreds. The appropriateness of their selection is seconded by decades of prior experience and data, as evidenced by the numerous articles referenced in this article.

For decades, we have learned, documented – and ignored – the same lessons:

- Attend to the individuals and their interactions.
- Put the mercilessness of running software to good use – write, and learn from the writing.
- Get inside the heads of your users and customers, and get them on your side.
- Plan coarse-grained long term and fine-grained short term, and find a way to replan quickly – because you'll have to.

And, finally:

- Do not believe in any single process or methodology because each works only in a particular and limited context.

Those who do not learn from history are doomed to repeat it. Let's stop pretending we don't know everything and let's stop repeating painful history by following this *good, old* advice.◆

References

1. Weinberg, G. The Psychology of Computer Programming. Dorset House, 1998.
2. DeMarco, T., and T. Lister. Peopleware. Dorset House, 1999.
3. Guindon, R., and B. Curtis. Control of Cognitive Processes During Software Design: What Tools Are Needed? Proc. of Chi '88 Conference: Human Factors in Computing Systems. Washington, D.C., 1988.
4. Naur, P. "Programming as Theory Building." Computing: A Human Activity. ACM Press, 1992.
5. Brooks, F. "No Silver Bullet." Computer Apr. 1987: 10-19.
6. Boehm, B. Software Cost Estimation With COCOMO II. Prentice Hall PTR, 2000.
7. Jones, Capers. Software Assessments, Benchmarks and Best Practices. Addison-Wesley, 2000.
8. Naur, P., and B. Randell, eds. "Software Engineering: Report on a Conference Sponsored By the NATO Science Committee." <http://homepages.cs.ncl.ac.uk/brian.randell/NATO>.
9. Allen, T. Managing the Flow of Technology. MIT Press, 1984.
10. Cockburn, A. Agile Software Development: The Cooperative Game. Addison-Wesley, 2006.
11. Olson, G., and J. Olson. "Distance Matters." Human-Computer Interaction. (2001): 139-179.
12. McCarthy, J., and A. Monk. "Channels, Conversation, Cooperation, and Relevance: All You Wanted to Know About Communication But Were Afraid to Ask." Collaborative Computing. Vol. 1, Mar. (1994): 35-61.
13. Hock, D. Birth of the Chaordic Age. Berret-Koehler, 1999.
14. Schwaber, K., and M. Beedle. Agile Software Development With Scrum. Prentice-Hall, 2001.
15. Royce, W. "Managing the Development of Large Software Systems." IEEE Wescon Aug. 1970: 1-9 <www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf>.
16. Cockburn, A. Surviving Object-Oriented Projects. Addison-Wesley, 1998.
17. Basili, Victor, and A. Turner. "Iterative Enhancement: A Practical Technique for Software Development." IEEE Transactions on Software Engineering. Dec. 1975: 390-396.
18. Larman, C. Agile and Iterative Development. Addison-Wesley, 2003.
19. Keil, M., and E. Carmel. "Customer-Developer Links." Communications of the ACM. May 1995: 33-44.
20. Laufer, A. "Managing Projects In a Dynamic Environment: Results-Focused Leadership" <http://appl.nasa.gov/ask/issues/19/19o_resources_letterfromtheeditor.php>.
21. Cockburn, A. "People and Methodologies in Software Development." Diss. University of Oslo, 2003 <http://Alistair.Cockburn.us>.
22. "Manifesto for Agile Software Development" <http://agilemanifesto.org>.

About the Author

Alistair Cockburn, Ph.D., is an expert on object-oriented (OO) design, software development methodologies, use cases, and project management. He is the author of "Agile Software Development," "Writing Effective Use Cases," and "Surviving OO Projects," and was one of the authors of the Agile Development Manifesto. Cockburn defined an early agile methodology for the IBM Consulting Group, served as special advisor to the Central Bank of Norway, and has worked for companies in several countries. More can be found online at <http://alistair.cockburn.us>.

1814 East Fort Douglas CIR
Salt Lake City, UT 84103
Phone: (801) 582-3162
E-mail: acockburn@aol.com

What Have 20 Years Accomplished and What Is Left to Accomplish in the Next 20 Years?©

Gerald M. Weinberg
Weinberg & Weinberg

Though there have been many improvements in software development in the last 20 years, our aspirations may have grown faster than our abilities. Despite improvements in hardware, tools, and processes, we still lag in our ability to spread the best management practices to more than a small fraction of organizations.

I was asked by CROSSTALK to help celebrate its 20 years of existence by writing an article on what has changed in the software business in the last 20 years. I would prefer to start with something that has not changed – people still mismanage software development at about the same rate they did 20 years ago. This figure may not be as pessimistic as it seems, because although we have made some improvements in the way software is developed, those improvements have encouraged people to undertake much more ambitious projects. We want systems that are faster (because we have faster chips), larger (because we have bigger storage devices), more connected (because we have a huge Internet), and easier to use (because we have seen quite a few easier-to-use systems throughout the past two decades).

Sadly, the number of programming languages and dialects of those languages continues to grow, though any improvements due to language changes come, at best, in small increments and people still write computer programs pretty much by hand. Whatever large quantities of juice that could be squeezed out of language improvements has dried up long ago, but computer science curricula still features compiler building as a major subject – to the exclusion of other subjects that might produce larger gains in productivity.

None of this affects me much because I no longer write code for a living. However, one of my sons writes code for a living. So, a generation has passed, but we are still building systems in much the same way we did a generation earlier (and a generation before that).

It is possible that neither of my grandsons will write code for a living because better electronic communication paths have enabled outsourcing jobs to lower wage areas around the world. My granddaughters may not either, and not just because of outsourcing. A generation ago, the percentage of women in the software development field was increasing, but now that number seems to be decreasing. (This is a personal observation based on the percentage of women participating in my workshops and conferences. For example, at a recent Amplifying Your Experience [AYE] conference, 15 out of 60 participants [25 percent] were women. That number would have been 40 percent 20 years ago. Similarly, my most recent two Problem Solving Leadership workshops each had eight women out of 24 participants, whereas 20 years ago, these workshops averaged 50 percent women.)

> *"It is a puzzle why so few software people use the information that is available to them; fewer, it seems, than 20 years ago ... one thing that has definitely changed is the instant availability of technical information on almost any topic."*

Why would the percentage of women be decreasing? Usually when wages in a field fall (as relative wages are now doing in software development), the percentage of females increases. Perhaps women have more opportunities elsewhere and are less tolerant than men of toxic work environments, as many software development environments can only be considered poisonous.

Management of software projects has improved in many places. If you want to be an optimist, you could say that half the large projects succeed to some extent. If you want to be a pessimist, you could say that half fail entirely. That is not much different from a generation ago. In absolute terms, the field is bigger, so we do have a larger number of successes, but the average manager's skill has not measurably improved in two decades.

Individual managers, however, have certainly improved. The good news is that we now have many more excellent managers who can mentor new managers. The bad news is that because the field has grown so much, there simply are not enough experienced managers to go around. We still see people abused or poorly used by their managers. All of this in spite of the fact that we have much more information about how to work with people in software development – information that, sadly, is frequently ignored. Too many managers have failed to learn that the world will not end due to a late delivery or a defect in production. Nor have they learned to relax, breathe, learn from their mistakes, and try again.

It is a puzzle why so few software people use the information that is available to them; fewer it seems, than 20 years ago. It is a puzzle because one thing that has definitely changed is the instant availability of technical information on almost any topic. Back then, mastering a new technology meant buying expensive books and accessing technical magazines on paper in remote places. Today, you can find the information in an instant (and for free) on the Internet (and sometimes it is actually accurate). So it is easier now to learn and improve, but only if you want to. It seems, though, that too few people want to take the trouble.

Perhaps software people want to improve, but lack guidance. Do we have

more guidance than we did 20 years ago? Well, one thing is for certain: We do have more PowerPoint. In fact, PowerPoint started almost exactly 20 years ago under the name of Presenter for the Mac. Microsoft purchased the product shortly thereafter and renamed it PowerPoint. For me and others like me who used great numbers of transparencies and 35mm slides, PowerPoint (and its clones) was a definite change for the better. Another change that has occurred is that these days I am making movies, such as one about dog agility that I did for an Agile conference keynote address, and I have also done several more that are available on YouTube[2]. Whether movies will be an improvement over slides remains to be seen.

But an even better change, for me at least, was getting rid of PowerPoint altogether and concentrating on experiential training. Some of us understand that you cannot learn software development from watching PowerPoint presentations. You have to get your hands on the actual software, which is much easier to do today. Contrast this with 50 years ago when I had to argue for weeks with IBM managers to allow my five-week programming class to run one program, one time, on an IBM 709 computer. That was one time for the entire group of 20 students. Today, I have four machines on my desktop, and each one is more powerful than that IBM 709.

Unfortunately, too many people still believe you can learn from PowerPoint presentations and then demonstrate your competence with software development or software management by passing tests. The misuse of certifications (for both the group and the individual) has grown enormously in 20 years, as executives wish (in vain) for some sort of assurance that their software projects will be successful. The number and size of organizations promising to fulfill this hope has significantly increased, but the hope remains unrealized.

The idea of certification is only one of the management myths that has not changed in a generation. We still have an excess of heroic environments, often leading to death march projects. Many managers still maintain the fallacious perception that quality can be tested into slapdash software. Some still try to develop software without being bothered by potential users of that software. And some still believe that process models are the answer to all of our problems.

These are the things I notice on my pessimistic days. On my optimistic days, I notice more managers realizing that it is the people who make a difference, that they can hire talent, but then must also build the relationships to build a team. Quite a few organizations are now using process models successfully, but as only one of the tools providing information for informed cultural changes to their software community.

Speaking of communities, I see the formation of more virtual communities of self-selected individuals such as open source development. Curiously, open source development is a (welcome) return to the way we practiced software development 50 years ago. Maybe we are not seeing change so much as we are seeing cycles.

"... along with incremental development, timeframes have changed. Everyone seems to be in a great hurry, not stopping to think as often and as deeply as developers did a generation ago. Or maybe this is just my perception as I grow older."

Speaking of cycles, we have learned that change is not easy for anyone, so it is best to proceed slowly and show incremental improvement. But, along with incremental development, timeframes have changed. Everyone seems to be in a great hurry, not stopping to think as often and as deeply as developers did a generation ago. Or maybe this is just my perception as I grow older.

These days, we do not seem to have the luxury of time, especially when we are releasing working software every two or four weeks. Sometimes this pace is good (no more *analysis paralysis*), but often it means producing a steady stream of technical debt due to quick, shallow thinking[1]. Throughout the decades, we certainly have accumulated an enormous technical debt in our software. We have just begun to realize the way technical debt slows down progress in the long run. If we are going to look better on CrossTalk's 40th anniversary, we are going to have to address some of these things that have not changed in the past 20 years.◆

Acknowledgement

I owe many of the ideas in this article to Charles Adams, Laurent Bossavit, Marjie Carmen, Dwayne Phillips, and Dave Smith from their contributions to the AYE conference wiki <www.ayeconference.com/wiki/scribble.cgi>.

Notes

1. See "Climbing Out of Technical Debt" by Johanna Rothman <www.ayeconference.com/climboutoftechnicaldebt>.
2. See <www.youtube.com/watch?v=77xrdj9yh3m> for more information and to see other videos on a variety of other software-related subjects by the author.

About the Author

Gerald (Jerry) M. Weinberg is a principal in the consulting and training firm Weinberg & Weinberg. For more than 50 years, he has worked on transforming software organizations. Weinberg is author or co-author of more than 40 books that cover all phases of the software life cycle, including, "The Psychology of Computer Programming," "An Introduction to General Systems Thinking," "Exploring Requirements," "Rethinking Systems Analysis and Design," "The Handbook of Walkthroughs," "General Principles of System Design," and "The Roundtable on Project Management." His books on leadership include "Becoming a Technical Leader," "The Secrets of Consulting," "The Roundtable on Technical Leadership," and the "Quality Software Management" series. Weinberg is also known for his conferences for software leaders, including the AYE conference.

www.geraldmweinberg.com

CrossTalk: The Long and Winding Road

Gary A. Petersen
Arrowpoint Solutions, Inc.

On CrossTalk's 20th anniversary, please join me as I open up the archives, pull back the curtain, and journey down the long and winding road of CrossTalk success. Discover the founders, the famous, the flippant, the firsts, the feckless, the facetious, and the fait accompli that is CrossTalk. Kick off your shoes, slide on your slippers, grab your favorite beverage, lay back, and enjoy the read.

In project management, milestones are marked by time and deliverables. In life, milestones are marked by events and beckoned with the wistful question, "Where were you when?" The attack on Pearl Harbor, the JFK assassination, the Apollo 11 moon landing, Woodstock, the Space Shuttle Columbia disaster, Chernobyl, the fall of the Berlin Wall, and the events of September 11 are examples of life's milestones. Whether tragic or euphoric, a milestone emerges from events of indelible emotional impact. In celebrating CrossTalk's 20th anniversary, the foremost question is, "Where were you when CrossTalk was first circulated?"

Get Back

Need help? It was July 22, 1988. There were nine pages in a two-column newsletter format. The publisher was Bill Frost, the editor was Susan Kelsey, and the main topics were Ada, artificial intelligence, and a Software Engineering Institute's (SEI) affiliates program. Ring a bell? Granted it was not a rock-your-world event, but a seed was sown that would yield future fruit. This article journeys through the CrossTalk archives to give you a behind-the-scenes look at your favorite publication set to a soundtrack by Lennon and McCartney.

Yesterday

First things first: Let's take a look at CrossTalk firsts. The first organization to publish CrossTalk was the Software Technology Office. CrossTalk's sister product, the Software Technology Conference, was first advertised in February 1989, and 200 people attended. The first obituary (the journal dabbled in obituaries in the early days) was Howard L. Yudkin in July 1989. The first mathematical equation appeared in October 1989 in an article on neural networks.

The March 1991 issue had an article on SEI's Process Maturity Model – the forerunner to the Capability Maturity Model®. That issue also had CrossTalk's first attempt at wit and humor with an article titled, "To Disc or not to Disk."

The January 1989 issue was the first funded by the Embedded Computer Resources Support Improvement Program, CrossTalk's sponsor for nearly 15 years, and later changed to the Computer Resources Support Improvement Program. The January 2004 issue was the first sponsored by multiple organizations. The Software Technology Support Center (STSC) currently reviews, edits, designs, and publishes CrossTalk.

> *"CrossTalk's intent has always been to entice software engineers and managers in the defense industry trenches to talk – thus, the name – and provide a forum to share software engineering experiences and insights."*

In an era when joint operations engendered lip service, CrossTalk reached across services to a broader defense community. Although it started in the Air Force, the first Navy article appeared in August 1991; NASA's first article appeared in June 1991; the first Army article in December 1993; the first Defense Information Systems Agency article in January 1994; and the Marines joined the party in October 1997. The June 1994 issue was the first official Department of Defense (DoD) issue.

The first double issue – one publisher's attempt to get back on schedule – was August 1991. The first interview, conducted with Dr. Delores Etter, appeared in July 1999. CrossTalk's Top 5 Projects Awards arrived in April 2001, and I am sure you are dying to know that this author's first byline surfaced in a feature column in February 1995.

A Day in the Life

CrossTalk's intent has always been to entice software engineers and managers in the defense industry trenches to talk – thus, the name – and provide a forum to share software engineering experiences and insights. CrossTalk was there for the young lieutenant saddled with Ada; for the chief information officer concerned about Y2K; for the new software engineering process group lead ordered to achieve Level 5; for the colonel who learned peer reviews offered him the best return on investment; and for the commander implementing net-centricity. All readers gained insight into the software development of the Lockheed Martin/ Boeing F-22 (October 1992) and Boeing 777 (January 1996). It certainly was ironic that Col. Gary Case provided a retrospective on the Integrated Computer Automated Software Environment (I-CASE).

With CrossTalk, you do not bleed on the edge of technology nor wade through the tall timbers of advertisements. It is concise, timely, applicable, and practical. It does not contain all the answers, but does help weary software soldiers on their way to a solution. Connecting software warriors continues to be the heart and soul of CrossTalk.

Help!

Although grounded in practicality, CrossTalk also knew it did not hurt to tap the industry's brightest brains.

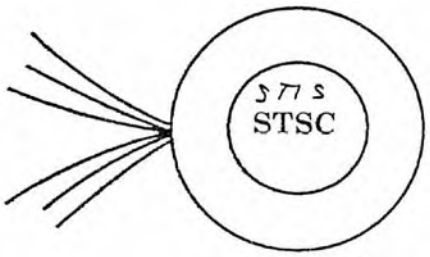

Figure 1: *The Original Logo*

Figure 2: *The STSC "Football" Logo*

CrossTalk has had its share of thinkers, in order of first appearance: Lloyd K. Mosemann II, W. Edwards Demming, Paul A. Strassmann, Capers Jones, Will Tracz, Tom DeMarco, Thomas McCabe, Elliot Chikosfsky, Grady Booch, Dave Cook, Watts Humphreys, Mark C. Paulk, Tom Gilb, Rayford B. Vaughan, Jr., Barry W. Boehm, Tim Lister, Alistair Cockburn, David Card, Gerald Weinberg, Bill Curtis, John Gilligan, Lt. Gen. Charlie Croom, and the Honorable John J. Grimes.

There have also been some atypical and dubious contributors. In August 1992, Tim Allen pontificated on Ada. It must have been before the gig with the tools. Darleen A. Druyun authored an article and no, it was not on the ethics of acquisition[1]. In the February 1989 issue, you could order the Computer Software Management and Information Center software catalog softbound ($25); 3.5" floppy disk ($30); 5.25" floppy disk ($30); nine-track magnetic tape ($50); or microfiche ($10). How ironic that the only media still viable today is the softbound copy. Hmmm ... might there be something to learn here?

Figure 3: *The* CrossTalk *"Stutter" Logo*

SOFTWARE TECHNOLOGY
SUPPORT CENTER
- - - CROSSTALK- - -
- - CROSSTALK- -
- CROSSTALK-

Figure 4: *The Logo Today*

CrossTalk
The Journal of Defense Software Engineering

Movie fans will be surprised to know Jason Bourne penned an article in June 1995 under his real name, David Webb (it must have been prior to his Treadstone assignment). Gandhi (Robin A., not Mohandas K.) offered enablers for software assurance – non-violent enablers, of course. Finally, my first article appeared in October 1995, and my first BackTalk article in June 1999; atypical, for sure. A $50 Ruth's Chris Steak House gift certificate awaits the first reader to e-mail me and tell me in which issue the author's picture appeared.

Let It Be

While substance is paramount in a professional journal, CrossTalk learned through the years that style and readability led to recognition, respect, and loyalty – branding for our business readers. CrossTalk style and format has truly taken a long and winding road.

The lanky nine-page, two-column newsletter had a bizarre logo in the upper right corner (Figure 1). I challenge anyone to tell me what the logo represents. I have no idea. In July 1989, an STSC football logo appeared – not my favorite (Figure 2), and in May 1990, CrossTalk developed a stutter (Figure 3). The then-publisher felt it necessary for the journal's name to be repeated three times in a Star Wars-style descending manner. The football logo came back in a special issue in 1993. This special issue was the only one to not have a monthly designation – it is a special issue just like the "White Album" is special. Finally, the familiar CrossTalk diamond logo you enjoy today made its debut in July 1996 (Figure 4).

> *"CrossTalk is the original shareware: free from ads and subscription cost – it was wiki before wiki was cool."*

Starting out white as the pure driven snow, in March 1991 CrossTalk switched to yellow paper. In January 1993, the yellow faded to cream. In 1996, the then-manager of the STSC, sent the staff on a search for the color *DoD purple*. Although it did not exist, they found it anyways and in July 1996, CrossTalk went DoD purple and white and eventually developed a purple stripe on its spine in May of 1997. Finally, the new millennium (actually July 2000) ushered in the full-color glossy cover you enjoy today.

The forerunner to the online edition of CrossTalk was the Electronic Bulletin Board System. At first, it did not contain CrossTalk online, just related material. In April 1990, CrossTalk was available online via file transfer protocol – pre-Internet. The STSC and CrossTalk went online in December 1992, and the current online version readers enjoy began in October 1997.

CrossTalk has matured from a humble newsletter to a technical report to a professional journal.

A Hard Day's Night

Behind the glossy cover are the managers, publishers, editors, and staff that make sure CrossTalk delivers. After the Frost-Kelsey duo, the vaudeville act of Alder-Klipper took over in January 1989 with Rudy Alder publishing and George Klipper writing and editing. What they lacked in resources and skill they made up for in vision and determination. November 1990 brought a new publisher in: Steve Wirick. The Wirick-Klipper team brought a bluegrass-roots feel to the publication as they expanded on Alder's basic groundwork.

In January 1992, Verlon Duncan replaced Wirick as publisher, Robert Bliss (Curmudgeon Bob) relieved Klipper, and in April of that year, John Fritschle joined as managing editor. The jazz trio worked their magic attracting a broader software audience as they infused topics outside the original embedded software sheet of music. Major Alan Miller expanded the trio to a quartet in June; performing as publisher and Duncan as manager.

In January 1994, Richard Smith replaced Duncan as manager, Major Pete Vaccaro replaced Miller, and Brent G. Wynn alternated with a returning Alder as associate publishers for the next two-and-a-half years. This group, like a marching band, was focused on efficiency and precision.

In July 1996, Rudy Alder assumed the manager's role and brought in a brilliant young publisher: Tracy Stauder. Together, they turned the marching band into an orchestra of substance, efficiency, and style. Heather Winward brought in the best authors, and Randy Schreifels and Mark Nielson took CrossTalk online.

In June 1999, the first professional editing staff was brought onboard, with Kathy Gurchiek as managing editor. Kathy was replaced by Pam Bowers in September 2003, then Kase Johnstun in February 2006. Currently, Drew Brown is the managing editor supported by Associate Editor Chelene Fortier and Article Coordinator Nicole Kentta.

Beth Starrett joined Tracy in November 2000 as an associate publisher and eventually replaced Tracy in March 2006 under the leadership of managing director Brent Baxter. Recently, Brent tapped Kasey Thompson to assume the publishing duties. All are commended for their contributions.

I Am the Walrus

All work and no play makes Jack a grumpy engineer. From the first foray into humor with "To Disc or Not to Disk," CROSSTALK found that laughter is the best medicine for the software blues. A decided effort at wit took root in March 1992 with Curmudgeon's Corner – the brainchild of Robert Bliss. Loren May took over in August 1996 and offered an outsider's view of the inside world of software engineering. In July 1997, Curmudgeon's Corner was renamed BACKTALK, which has been a mainstay ever since.

Since June 1999, BACKTALK has been composed by the professor (Dr. Dave Cook) and the madman (yours truly) with occasional guest writers when the publisher was mad at the two of us. For rookie readers, the proper way to read CROSSTALK is to start with BACKTALK. It warms up the mind in preparation for the technical feast inside. To quote Tracy Stauder, "BACKTALK [has] always been a favorite – engineers need humor!" I say most engineers have a screw loose that needs a droll tweaking once in a while.

Come Together

CROSSTALK is the original shareware: free from ads and subscription cost – it was wiki before wiki was cool.

Like shareware, CROSSTALK depends on your contributions to stay alive. The easiest contribution is your feedback to the publisher on how CROSSTALK has helped improve your software engineering efforts, saved your project money, or boosted product quality. I cannot tell you how valuable this is to CROSSTALK, the staff, and you the reader. Please take time to impart CROSSTALK's contribution to your success by e-mailing the publisher at <stsc.customerservice@hill.af.mil>.

You can contribute an article on your victories (what to do), your failures (what not to do), or your experience as a software engineer or manager (what I did).

Finally, if you are a successful, innovative leader in the industry, please consider sponsoring an issue or two per year. You know what they say: Those who can, do. Those who cannot, teach. And those who care, sponsor CROSSTALK. Make your contribution count. Be a part of the next 20 years.

Here, then, is a toast to the dreamers and schemers, the movers and shakers, the editors and publishers, the authors and sponsors and especially to you, the writers and readers: to CROSSTALK – may the next 20 years outshine the last 20.

"I get by with a little help from my friends."◆

Note

1. See <http://en.wikipedia.org/wiki/Darleen_Druyun> for more information.

About the Author

Gary A. Petersen is the Executive Vice President of Arrowpoint Solutions, Inc., where he develops new business opportunities and manages small teams of software engineers, IT professionals, instructional designers, and the editing staff of CROSSTALK. His experience includes six years as a lead software engineer designing, developing, and testing operational flight software for the Air Force F-16 Fighter Jet, seven years as a technical lead and manager of the Software Technology Support Center, one year in automated testing purgatory, and 10 years developing small businesses to serve the best customers on planet Earth. Petersen can still put his socks on without sitting down, is addicted to Coca-Cola, and frequently contributes BACKTALK articles where he ensnares, fillets, and displays his inner child on garnish-encircled ice.

129 South State
STE #100
Clearfield, UT 84015
Phone: (801) 773-8822
E-mail: gpetersen@arrowpoint.us

COMING EVENTS

September 9-11
AIAA Space 2008 Conference & Exposition
San Diego, CA
www.aiaa.org

September 10-11
7th Annual QAI & QAAM Regional Conference
Baltimore, MD
www.qaiworldwide.org/conferences/qaam08/index.html

September 15-18
4th World Congress for Software Quality
Bethesda, MD
www.asq.org/conferences/wcsq

September 22-25
3rd Annual Software Engineering Institute Team Software Process Symposium
Phoenix, AZ
www.sei.cmu.edu/tsp/symposium.html

September 26
Making the Business Case for Software Assurance
Pittsburgh, PA
https://buildsecurityin.us-cert.gov/daisy/bsi/events/972-BSI.html

September 29–October 3
STARWEST 2008
Software Testing Analysis & Review
Anaheim, CA
www.sqe.com/starwest

October 14-16
Software Assurance Forum
Gaithersburg, MD
https://buildsecurityin.us-cert.gov/daisy/bsi/events/930-BSI.html

2009
2009 Systems and Software Technology Conference
Salt Lake City, UT
www.sstc-online.org

COMING EVENTS: **Please submit coming events that are of interest to our readers at least 90 days before registration. E-mail announcements to: nicole.kentta@hill.af.mil.**

I FIND CrossTalk [IS] A USEFUL SOURCE OF TIMELY AND ACCURATE INFORMATION N
◆ I CAN THINK OF NO FINER COMPLIMENT THAN TO SAY THAT IF I HAD TO KEEP ONLY
EMPLOYEES AND I LOOK FORWARD TO EVERY ISSUE, AND ITS ARTICLES ARE SUBJECTS O
TREMENDOUS VALUE THAT I PLACE ON YOUR MAGAZINE. *—Robert Miller, Director of Sof*
SCHEDULE PRACTICES THAT WALT LIPKE HAS BEEN WRITING ABOUT FOR YEARS. ALSO, I'
PROCESS AND PERSONAL SOFTWARE PROCESS. IN ADDITION, WE'VE MADE SOME CHA
ALL, THESE CHANGES HAVE BEEN KEY IN PICKING UP A $20M PER YEAR WORKLOAD AND
[YOUR] MAGAZINE CERTAINLY HAD KEY IMPACTS. *—David R. Webb, Senior Technical Prog*
BEST PRACTICES IN SOFTWARE DEVELOPMENT AND, MORE GENERALLY, FOR SYSTEMS DE
TO GET A GOOD UNDERSTANDING OF THE TOPIC BUT IS AT A SUFFICIENTLY HIGH LEVEL
VICE TO THE COMMUNITY. *—Paul Davis, NGC* ◆ TIME. MONEY. PROCESS ... CrossTalk
—Peter Maclean, Technical Architect, DND Canada ◆ I'VE BEEN USING CrossTalk SI
TINUE TO BENEFIT FROM IT TODAY ... THE METHODOLOGIES THAT UNDERPIN SOFTWAR
JECT MANAGEMENT) ARE MOST HELPFUL IN MY DAY-TO-DAY OPERATIONS. PLEASE CO
Aldermaston, UNITED KINGDOM ◆ I OFTEN USE PROCESS-RELATED ARTICLES FROM Cro
TO BE ABLE TO QUOTE EXAMPLES FROM ORGANIZATIONS WITH SIMILAR MISSION-CRITI
NOLOGY" PERIODICALS. MANY OF THESE ARE AN ATTEMPT TO SELL SOMETHING OR ARE
AS MY FAVORITE PERIODICAL. I CAN'T BEGIN TO TELL YOU HOW MANY CHANGES I HAVE
IS FILLED WITH "HOW-TO" INFORMATION THAT CAN BE APPLIED WITH MINIMAL EFFORT.
OUTSTANDING WORK. *—Steve Devinney, Software Quality Assurance Process Engineer, H*
B-1B PROGRAM, I WENT INTO SOFTWARE BECAUSE OF THE LACK OF UNDERSTANDING I
WITH THE COMPANY'S SYSTEMS ENGINEERING MANAGER AND THE SOFTWARE MANAGE
LY DECIDED TO HAVE MEETINGS IN THEIR MANAGER'S OFFICE AND TALKED TOGETHER! W
THEY ARE GRATEFUL FOR BECAUSE IT HELPS THEM KEEP CURRENT ON INDUSTRY THINK
LEYAN UNIVERSITY, WE HAVE AN UNDERGRADUATE DEGREE IN SOFTWARE ENGINEE
CrossTalk GIVES ME A REALITY CHECK TO STAY ABREAST ON WHAT THE CURRENT P
DENTS APPRECIATE SEEING CURRENT INFORMATION THAT CROSS-CHECKS WITH WHAT T
I HAVE BEEN A SUBSCRIBER TO CrossTalk SINCE THE DAYS OF PAPER IN THE EARLY
ARTICLES PROVIDED. WHILE MOST I.T. JOURNALS ARE GEARED TOWARD EITHER MARK
PRACTICAL, ACTIONABLE INFORMATION THAT IS USEFUL IN THE REAL WORLD ... THE IN
JOURNALS. *—Brian Spaulding, System Architect, CCM Professional Services, Pitney Bowes S*
ERENCE BASE FOR THE PROJECT MANAGEMENT INDUSTRY. *—Dr. Gary L. Richardson, Pro*
SAVAGE TECHNICAL WORLD OF BITS AND BYTES, AND IT ALWAYS AMAZES ME HOW A F
COMPUTER CAN EVER PROVIDE. MACHINES ARE OUR TOOLS, BUT PEOPLE ARE OUR BUSI
Computer Scientist, Naval Surface Warfare Center, Dahlgren Division ◆

Happy 20th Anniver

NOT FOUND ANYWHERE ELSE. —*Claire W. Jones, Software Engineer, The Boeing Company*
Y ONE PUBLICATION TO READ EACH MONTH, CROSSTALK WOULD BE THAT ONE. MY
F MANY GREAT DISCUSSIONS ... IT IS DEFINITELY MY PLEASURE TO SHARE WITH YOU THE
ftware Engineering, Data Device Corporation ◆ I HAVE BEEN IMPLEMENTING THE EARNED
'VE USED CROSSTALK SEVERAL TIMES FOR BEST PRACTICES ON THE TEAM SOFTWARE
NGES TO TSP/PSP BASED UPON AGILE PRACTICES THAT HAVE BEEN BENEFICIAL. OVER-
O IMPROVING PRODUCTIVITY ON A PROJECT BY 4:1 ... THE INFORMATION I GAINED FROM
gram Manager, 520 SMXG ◆ I USE CROSSTALK ON A REGULAR BASIS FOR IDENTIFYING
EVELOPMENT. I FIND THAT THE ARTICLES ARE AT THE RIGHT LEVEL OF TECHNICAL DEPTH
L TO GET A RAPID UNDERSTANDING OF ITS APPLICABILITY. KEEP UP THE VALUABLE SER-
K SAVES MY OFFICE BOTH OF THE FIRST TWO BY HELPING US IMPROVE THE THIRD ONE.
NCE I WAS INTRODUCED TO IT IN 1998 ... I FOUND IT OF GREAT VALUE THEN AND CON-
RE PROGRAMS AS PRESENTED WITHIN CROSSTALK EACH MONTH (CMMI, LEAN, PRO-
ONTINUE PRODUCING THIS EXCELLENT PUBLICATION. —*Charles Fraser-Fleming, AWE plc*
OSSTALK TO SUPPORT RECOMMENDED PROCESS IMPROVEMENT INITIATIVES. IT HELPS
ICAL CONCERNS. —*Huet Landry, Requirements Analyst, Unisys* ◆ I RECEIVE MANY "TECH-
STEEPED IN THEORY AND COMPLEXITY. OVER THE YEARS, CROSSTALK HAS EMERGED
IMPLEMENTED AS A RESULT OF AN ISSUE OR ARTICLE. I FIND THAT YOUR PUBLICATION
. IT'S REAL WORLD, TRIED, AND PROVEN METHODS RATHER THAN THEORY. KEEP UP THE
Harland Financial Solutions ◆ WHEN I STEPPED DOWN FROM THE PI POSITION ON THE
FOUND BETWEEN SYSTEMS ENGINEERING AND SOFTWARE. DUE TO MY RELATIONSHIP
ER AND WITH THE HELP OF CROSSTALK ARTICLES THAT I SENT TO THEM, THEY FINAL-
WHAT A MILESTONE THAT WAS! AND I STILL SEND CROSSTALK TO THESE GUYS, WHICH
KING. —*Melonee Moses, System Integration Facility Lead, DCMA* ◆ HERE AT INDIANA WES-
RING, AND I THINK IT'S FAIR TO SAY OUR PROGRAM IS STILL IN ITS INFANCY. I FIND
PRACTICES ARE. I FREQUENTLY MAKE REFERENCE TO [IT] IN A CLASS SESSION ... THE STU-
THEY FIND IN THEIR TEXTBOOKS. —*Bill Cupp, LCDR, USN (Ret.), Associate Professor, IWU* ◆
'90s AND HAVE BEEN HIGHLY IMPRESSED WITH THE QUALITY AND TIMELINESS OF THE
ETING OR ACADEMIC THEORY, CROSSTALK IS ONE OF THE VERY FEW THAT PROVIDE
FORMATION DENSITY STANDS IN STARK CONTRAST TO THE FROTH THAT ADORNS MOST
oftware ◆ I VIEW CROSSTALK AS THE MOST TECHNICALLY VALID AND AVAILABLE REF-
oject Management Certificate Program Coordinator, University of Houston ◆ WE LIVE IN A
FEW PIECES OF PAPER CAN REMIND ME THAT THERE IS MORE TO MY JOB THAN WHAT A
NESS. MY THANKS TO ALL OF YOU FOR MAKING THAT CLEARER TO ME. —*Stephen Voynar,*

sary CROSSTALK

Wiki Customization to Resolve Management Issues in Distributed Software Projects

Jamie Hohman
Garmin International

Dr. Hossein Saiedian
University of Kansas

This article introduces methods of wiki modification that can assist software project managers in addressing the communication problems associated with distributed projects. It outlines some of the best practices from industrial projects that employ distributed teams and discusses how such practices can be implemented via the wiki processes.

Software development projects are constantly becoming more complex. As customers continue to shift more of the risk to the development organization, project managers search for ways to mitigate this risk while still providing the requested functionality in a timely manner. Many managers increase the size and capability of their development organization by expanding teams across sites and even across continents. These distributed project teams introduce a new set of challenges to traditional project management.

Most of these challenges involve the level of communication that grows nonlinearly compared to the increase in project team size. The addition of geographically remote teams that may span time zones, cultures, and languages complicates communication even further. Effective tools are necessary to make this communication overhead manageable.

One communication tool which has been around for over a decade is a wiki. Wiki means *quick* in Hawaiian and consists of two things: 1) a program that makes editing Web pages extremely quick and easy, and 2) a philosophy regarding how users should go about that editing [1]. The wiki way philosophy allows any user to edit any page created by another user with the belief that fixing incorrect content is easier than painstakingly ensuring content is always correct the first time. Companies as diverse as Motorola and SAP are already using wikis in their day-to-day business operations[1]. Furthermore, there is already an annual international symposium on wikis and in particular, their use in software engineering and management[2]. The site highlights a quote by Shashi Seth, senior product manager at Google: "This company runs on wikis," referring to the uses of wiki tools such as GooWiki TWiki, and Sparrow.

This article focuses on the use of wikis in project management and, in particular, as a tool for project development oversight and as a communication tool by project management personnel and development team leaders.

Wikis have been in use for some time as a project management tool, but the very freedom and flexibility that make them powerful also make them difficult for large scale use. With some simple guidelines and modifications, this power can be harnessed. For a wiki to serve as an effective central knowledge repository, it must be easily searchable. Adding semantic information in the form of metadata tags is important.

The most powerful aspect of wikis is their flexibility, but project management and software development processes require some level of discipline. Adding light constraints to the wiki helps provide guidance for team members and improves monitoring of project status by management.

Distributed Project Management Challenges and Current Solutions

Software Project History and Evolution

As steady hardware development has produced more machine capability, solutions to more complex problems are desired that utilize this increased hardware capability. These complex solutions tend to exceed the customers' technical knowledge, so their focus shifts from requirements to objective capability. Projects without clear, precise requirements increase the risk for the developer. The increased complexity of the solution often requires development of a system of systems (i.e., an enterprise) rather than one isolated system. Ensuring that these systems all communicate and operate together correctly increases the technical complexity of whatever solution is designed. To design and build these large, complex solutions in a timeframe required by the customer, the project manager must often utilize strategic teaming, outsourcing, and acquisitions [2].

Just as the software development community was starting to come to grips with the *software crisis* of the 1980s and early 1990s, the Internet opened previously unheard of communication channels. These new channels were quickly leveraged to build new distributed systems that created their own unique set of management challenges. Development teams soon realized this new, unique form of communication could also be utilized to create distributed project teams with significant advantages over traditional teams as shown in the following [2]:

1. Creating multiple, independently located teams results in a larger workforce than can be contained in any single location.
2. With teams around the world, time zone differences can be utilized to increase the number of productive hours during the day. Scarce or expensive machines (or other non-human resources) can be utilized at a higher rate. Artifacts produced by one team can be consumed by another team within the same day rather than waiting until the next day.
3. Knowledge experts and other specialized resources can be utilized regardless of where they reside.
4. Less expensive labor can be utilized in some geographic regions without experiencing a drop-off in expertise or productivity.

Since team members of distributed projects are no longer necessarily part of one large, homogenous group, some new development issues arise [2]:

1. It becomes difficult to leverage and track available resources. Even when a functioning system is in place, it can be time consuming to maintain.
2. Different organizations have different processes and capabilities so estimating effort and synchronizing work between sites is difficult.
3. The Internet allows communication and collaboration at levels never experienced before, but this communication is always less rich than face-to-

face communication, increasing the risk of miscommunication. The simplicity of Internet communication, however, also increases the volume of communication between team members which can lead to overload.
4. Different sites have different cultures so soft skills of employees and managers must be changed to conform to one standard of behaviors, processes, and technologies. Trust must be established at these different sites in the new processes as well as with other team members with whom they have not worked before.
5. Transmitting all of this knowledge over an inherently open system requires some additional security policies to protect the confidentiality and privacy of customer information.

A combination of the increased size and complexity of the system to be developed, as well as the new challenges listed that come with managing a distributed project, may make it extremely difficult for a project manager to control and coordinate all of the activities associated with modern software development projects. This necessitates a shift in his/her work away from the role of a benevolent dictator in a top-down hierarchical structure and develop new skills as an orchestrator of interconnected relationships [2]. Wikis are helpful tools for this orchestration. In fact, wikis provide such a great degree of openness, tolerance, and empowerment of the individual, that the previous dictatorship model would be nearly impossible to achieve. By their very nature, wikis delegate control and promote cooperation based on trust [1], forcing a more democratic approach to project management that is necessary in a distributed environment.

Current Distributed Project Management Methods

Current research in the field of distributed project management has outlined some key areas of focus that should be addressed to lead to more successful distributed projects. First, the analysis and effort that goes into creating project artifacts (such as effort estimation, resource allocation charts, and decision rationale) is as important to keep as the artifact itself. Next, the communication between team members during the design and development phases must be stored and available to everyone on the teams for future reference. Also, management and other team members must have visibility into the status and progress of development activities. This allows peer review at all levels as well as more accurate progress tracking. Every project should also have an electronic project repository where all relevant project information is stored to eliminate data re-entry costs, eliminate duplication and information loss, standardize work processes, and increase management's ability to efficiently retrieve accurate information leading to greater management control [3]. A collaborative project management architecture that could be implemented using a wiki is also proposed.

A suggested number of practices useful in increasing collaboration during distributed projects include synchronization of main milestones (more granular is unnecessary), frequent deliveries to create visibility in the project, establishment of peer-to-peer links to eliminate management bottlenecks, problem solving practices (especially at the project planning phase), informing and monitoring practices for management and employees, and relationship building practices to establish trust between teams [4].

Wiki Usage in Collocated Project Management

Wikis are already in use in project management roles, particularly in the open source community. The flexible nature of wikis allows use across the project management activity spectrum including project selection, cost estimation, effort estimation, scheduling, and progress tracking. They are also used, among others, for the following activity-independent tasks [5]:

- Creation and storage of deliverables.
- Meeting agendas and outcomes.
- Status reports.
- *Great ideas* saved for later.
- Standard and best practices.
- New and innovative methods and processes.
- Recording claims for usage of shared machines and other resources.
- Vacation schedules[3].

Current research shows that these wikis are standard, completely open repositories with little in the way of guidance for the users. While it has been demonstrated that wikis are sustainable and increases are shown in contributions and accesses as they mature, the true effectiveness of wikis will be demonstrated as greater numbers of users are obtained [5]. It has also been shown that a dedicated administrator will decrease the apparent chaos of the wiki and lead to greater organization [6]. They also believe that user training is essential for a well-structured, organized wiki. If a wiki has support from a methodology advocate and from upper management, the collaborators will be more effective in creating and using content. As will be shown in later sections, light constraints imposed on the wiki will allow user training and methodology conformance to occur informally and continuously.

Wiki Modification: Recommended Practices

Standard usage of wikis is already common in project management. The ad-hoc nature of wiki development is one of the most powerful aspects of wiki usage. When supplemented by a few standard project management wiki items, the wiki becomes even more useful.

Rapid Feedback Areas

The project management wiki should provide areas for quick turnaround on evaluation of work products, including integration and testing reports, code reviews, and documentation approval. This keeps the process and artifact clear in the developer's mind and provides enhanced motivation. Both management and peers provide this evaluation of the work products. As the manager sees that the developer is doing good work (via these evaluations), the manager starts to trust and respect the developer's abilities [4]. Peer evaluations breed trust between co-workers, which is difficult to develop in a distributed project environment; efforts should be made to enhance trust whenever possible.

Peer-to-Peer Links

Peer-to-peer links should be established so each level of the project teams know who to contact on remote teams and what is the easiest method of contact. These links should be established at three levels: upper management, project managers, and developers [4]. As project teams grow, proper communication of all relevant knowledge becomes more burdensome. One way to help ease this burden is to eliminate redundant and misdirected communication. If the message gets to the right person the first time, communication burdens are eased.

Role Descriptions

The wiki should include role descriptions for enhanced communication between personnel that are not necessarily peers. A vital role here is *problem solver*. A problem solver is a highly experienced team member who either knows the answer to the problem or knows where to find it. There should be one on each team, and

everyone on other teams should know who that person is. If this role goes unfulfilled, team members may spend more time trying to find someone to help them on a remote team than they would actually solving the problem [4]. These types of problems can build to disastrous proportions at the end of projects when time constraints are most critical. It is also important for project managers to know who the designated problem solver is on the team because answering these questions can take a large amount of this individual's time. The project manager should schedule accordingly.

High-Level Reports

High-level project status and progress reports should be included on the wiki. This provides a window for both upper management and developers to recognize priorities and gain motivation from a project that is scheduled well. If the communication process in other areas breaks down, these high-level reports can serve as a backup so that remote teams have a chance to learn of changes or decisions made to a project. They can also serve notice that artifacts required for further project work are now available [4]. For example, if a development team is waiting on a requirements specification from a business analyst to begin their development work, the development team would be able to see that the business analyst has posted his work on his wiki page and they can begin working on it sooner.

Face List

To further build the trust level between remote teams, face lists and organization charts should be posted. Knowing who someone is, what their role is in the organization, and receiving feedback from that person creates trust and eliminates the feeling that remote team members are unknown and easily disregarded partners [4]. This also encourages more targeted communication, which eases the burden on project managers and problem solvers who often become communication bottlenecks. It must be noted that not all organizations, especially those within the military and intelligence communities, would want to provide as much detail and identifying information for their personnel and organizational structures.

Project Dictionary

A project dictionary where key terms, jargon, and methodology are defined is useful. This benefits both developers (for enhanced clarity) and other stakeholders who may not possess as much technical knowledge (i.e., management, sales, etc.) [3]. As this dictionary matures it will provide project context, background, and boundaries that would otherwise be lost as memories fade.

Business Rules and Policies

Project members should post a list of business rules and policies that are not specified in more formal documents or a link to these documents. This leads to enhanced standardization across all sites and quicker adaptation by new team members [3].

Semantic Wikis

The combination of ad-hoc information contributed by team members and the project management information contributed by leadership produces a large amount of data that most readers have neither the ability nor the motivation to search through while trying to complete their day-to-day work. Adding semantic data to wiki entries can make this process easier.

Adding metadata to wiki pages is most effective when the granularity of the metadata is chosen carefully. All of the pages should include a vertical metadata tag, a horizontal metadata tag, or both. The vertical tag would indicate a *describes-feature* relationship. For example, the notes from a meeting that planned the overall database schema of a project would have a describes-feature property that relates to database schema. The horizontal tag would indicate what step in the project management plan this document pertains to, or a *plan-level* tag. For example, a Gantt chart that shows the resource allocation plan would have a plan-level tag that relates to scheduling. The schema planning meeting notes described above would also have a plan-level tag that relates to architecture planning. One flavor of this idea is in use in Wikipedia where typed links are highly encouraged. Typed links means that the author gives information about the link while creating it. For example, when creating a link from Germany to Berlin, the link is typed as: *is-capital-of*.

The second level of granularity for metadata tags would be chosen by the individual page authors conforming to the wiki way. As a team works on designing or implementing a certain feature, the team members decide what aspects of their documents in the wiki they want to highlight and add metadata tags consistently and appropriately. Good communication would allow ideas to naturally cross-pollinate between teams as best practices emerge. This would lead to some level of uniformity across the organization.

One important aspect of adding semantic data to wikis is that these additions should *always be left as optional*. One aspect of what makes wikis powerful is the naturally low entry barrier. Any requirements that raise this barrier will discourage participation and should be rejected [7].

As all of this metadata is collected in the wiki, it must be able to be queried or it will not be useful. Simple queries that use the suggested describes-feature and plan-level tags should be readily available. Another mechanism that evaluates arbitrarily complex queries should also be available for the advanced user. Two other methods to easily formulate queries of medium complexity are provided in [7]. One is keeping track of the links the user traverses during normal navigation and prompting with these types when entering the query. The other is taking feedback from the user concerning query results to further refine the search.

Light Constraints on Wikis

A method of adding useful structure to a wiki without constricting the features that make wikis so powerful, namely the low entry barrier and tremendous flexibility, is outlined in [8]. Even though wikis are tremendously flexible, it is generally desirable to have a structure that is enforced throughout the wiki to maintain a consistent look and feel throughout the wiki as well as forcing similar pages to have similar content and structure. In the case of a project management wiki, the management process may require certain sections and content, for example, a section entitled "Feasibility Study" or one entitled "Nominal Costs," or one entitled "Effort Multipliers/Drivers" to be included in all cost estimation pages. In traditional wikis, the maintenance of this structure is accomplished through copy-and-paste creation, manual refinement, and checking by WikiGnomes (editors who work quietly to fix minor nuisances). To begin to automate this process, a proposed set of rules enforced by the wiki engine points out to editors and viewers when something about the content of the page is unacceptable [8]. This can be something as simple as violating the HTML or the company-wide cascading style sheet (CSS) rules, or introducing an acronym for the first time without providing appropriate references for it. These rules are guidelines that the wiki page ought to satisfy. They are generally identified by recognizing properties that

pages of the same type should share.

Complete freedom from editing is retained in two ways. First, any page can be saved and viewed at any time, even if it does not abide by all of the rules set forth for that page. However, the rules broken will be illustrated on that page at all times, both to the editor and the viewer. If an editor attempts to save a page that is breaking these rules, the system should issue a prompt before allowing the save to verify that these rules are going unfulfilled. Secondly, any of these rules can be added, modified, or deleted at any time. If so chosen, the wiki administrator could set up a group of *tailors*. The tailors would be the only users allowed to edit the rules (e.g., the CSS rules). This would provide for heavier constraints which may be desirable in a specific wiki.

Adding light constraints to wiki entries helps the project manager in multiple ways while still keeping the traditional benefits of the wiki way:

1. Light constraints allow the project management authority to specify and modify standard formats for required project artifacts that are stored in the wiki. If these standards change, the project managers need to only make these changes in one place, and the people responsible for these artifacts will automatically see that they need to be updated.
2. Light constraints allow the author of the artifacts to easily see what aspect of their artifact is unacceptable, yet still be able to save it and work on it later. This removes many of the major inconsistencies between artifacts developed by different people while alleviating the burden of standards enforcement from project managers.
3. These constraints allow the project manager to view the status of required artifacts. This will almost be in real time. When a standard format must be modified, the status of every artifact that uses this standard will automatically be updated by the wiki add-in software that enforces the light constraints.
4. These constraints should be tailored so that each page is required to have appropriate metadata tags as described previously. Without having some way to remind authors that these tags are important, it would be easy to overlook their addition and maintenance.

Light constraints are already in use in practice to help spell-check wiki content. Two examples of this are tools named MoinMoin and DokuWiki. Also, a project called Miki is attempting to make an infrastructure for collaborative authoring of formalized mathematics. Light constraints are used to encode consistency of mathematical concepts [8].

Motivations for Wiki Use

Even if a first-class wiki is designed, if it is not accepted by the project team or is viewed as potential overhead that takes time away from core project work, the community will not build it and the design effort is not worthwhile. Team members must be presented with factors such as the following to motivate them to use the wiki:

1. Encourage posting of vacation schedules, best practices, innovative ideas, great ideas saved for later, etc. Each team member could make their own section and personalize it with information about that person in which the rest of the team would be interested.
2. Archive absolutely *all* information. Include meeting agendas, meeting notes, formal status reports, all information that led to a decision that was made, and real-time capturing of meeting information. Eventually, memories fade and important project-related information will be lost.
3. Project authorities must use wikis, encourage others to use them, and demonstrate successes. Leaders should point out to team members that if the requested information was on the wiki, everyone's communication overhead would be lessened. As more content is added, the utility of the wiki will become more pronounced, starting a cycle of adding more useful content to a useful system.
4. Recognize personality types who are *adders* versus those who are *synthesizers* and encourage each person to pursue that direction. Adders are utilitarian and are more concerned with efficiently finishing the work by adding content rather than their reputation or task novelty. Synthesizers are more interested in impacting their tasks by finding a novel solution and enhancing their reputation by synthesizing already existing content [5].
5. Provide a mobile, device-friendly interface. Team members on distributed projects travel frequently and richly enabled mobile devices are becoming common. The downtime provided while traveling (such as at airports and in hotel rooms) provides a great opportunity to catch up with the status of other areas of the project that they may not be familiar with. Formal documentation does not translate to mobile devices easily and it is often out of date.

"High-level project status and progress reports should be included on the wiki. This provides a window for both upper management and developers to recognize priorities and gain motivation from a project that is scheduled well."

Wikis Serve a Distributed Project Management Need

Clearly, the manner in which current distributed project management is approached can be improved. Most of these deficiencies center around the challenges of effective communication between geographically separated project teams. Many project managers seek communication systems that support real-time, multimedia collaboration. This is not useful for developers who generally work alone. Also, as teams spread across continents and even across multiple continents, time zone differences between team members often make real-time collaboration unwieldy. The use of wikis as project management tools resolves many of these issues.

Impromptu, casual communication (such as e-mail, telephone calls, and Internet chat) is critical and can generate great amounts of high-value information, but that information is lost as soon as the telephone call or chat session is ended. That information is also never successfully disseminated across the project team if it remains in one person's private e-mail box or chat log. Semantic wikis help organize this extreme amount of information and facilitate searching it so more knowledge is present and readily available to those who need it. Oftentimes, knowl-

edge management systems and expert systems provide this service. Unfortunately, they are generally costly to implement and maintain, have a high entry barrier for first-time users, and are not well suited to span geographically remote locations. None of these restrictions exist in wikis.

The flexible nature of wikis allows project managers to shape and monitor the flow of information among distributed team members (for example, for the presence of certain page headings, preamble, or prologue information, in a particular color and font size, and within a predefined line length that would enhance readability and comprehension). Adding light constraints to the system will automate some of this monitoring without restricting the flow. It will also increase conformity to standards as the wiki can demonstrate where a critical document is insufficient.

Faster problem solving through both wiki content about the project as well as information posted on the wiki about whom to contact on remote teams will alleviate what is often a major communication bottleneck in project work. It will also alleviate work for the designated problem solver on each team who is usually an experienced team member and whose time is very valuable. As problems are solved more efficiently across teams, greater trust and reliance will be created between teams [8, 9, 10].

Conclusion

This article has outlined a set of useful pages that should be included in distributed project management wikis by leadership and grown by the team. They include areas used to communicate quickly between developers and management, areas used to communicate between peers on distributed projects to increase trust and productivity, and areas that provide increased visibility on project status and progress to all stakeholders. Adding semantic metadata in both the vertical and horizontal direction to this large repository helps to efficiently search and maintain it. Light constraints on the wiki aids project managers in their quest for conformity (when necessary) and status monitoring without increasing the entry barrier to the wiki and, thereby, discouraging use. Like other communication tools, modified wikis will not be adopted by the team at-large without concerted management support and demonstrated effectiveness. Once this buy-in is achieved, the wiki will organically grow and mature into an indispensable tool.◆

References

1. Louridas, P. "Using Wikis in Software Development." IEEE Software Mar./Apr. 2006: 88-91.
2. Nidiffer, K.E., and D. Dolan. "Evolving Distributed Project Management." IEEE Software Sept./Oct. 2005: 63-72.
3. Chen, F., J.F. Nunamaker, N.C. Romano, and R.O. Briggs. A Collaborative Project Management Architecture. Proc. of the 37th Annual Hawaii International Conference on System Sciences, 2003: 12-24.
4. Paasivaara, M., and C. Lassenius. "Collaboration Practices In Global Inter-Organizational Software Development Projects." Software Process: Improvement and Practice Sept. 2004: 183-199.
5. Majchrzak, A., C. Wagner, and D. Yates. Corporate Wiki Users: Results of a Survey. Proc. of the Association of Computing Machinery (ACM) International Symposium on Wikis. Odense, Denmark: 2006.
6. Raygan, R.E., and D.G. Green. Internet Collaboration: Twiki. Proc. of the 2002 Institute for Electrical and Electronic Engineers Southeast Conference. Louisville, KY.
7. Kiesel, M., and L. Sauermann. "Towards Semantic Desktop Wikis." Upgrade: The European Journal for the Informatics Professional 6.6 (2005): 30-34.
8. Di Iorio, A., and S. Zacchiroli. Constrained Wiki: An Oxymoron? Proc. of the ACM International Symposium on Wikis, Odense, Denmark: 2006.
9. Da Lio, E., L. Fraboni, and T. Leo. Twiki-Based Facilitation In a Newly Formed Academic Community of Practice. Proc. of the ACM International Symposium on Wikis, San Diego, CA: 2005.
10. Haake, A., S. Lukosh, and T. Schummer. Wiki Templates: Adding Structure Support to Wikis On Demand. Proc. of the ACM International Symposium on Wikis, San Diego, CA: 2005.

Notes

1. See <http://twiki.org> for details.
2. See <www.wikisym.org>.
3. This may affect privacy concerns for some employees. It is a suggestion only for organization where employees do not object to publishing their schedules.

About the Authors

Jamie Hohman works as a software engineer for Garmin International and has previously worked as a software development consultant in Kansas City, Kansas. Hohman is currently pursuing his master's degree in computer science from the University of Kansas. He received his bachelor's degree in computer science from Kansas State University in 1997.

Garmin International, Inc.
1200 E 151st ST
Olathe, KS 66062
Phone: (913) 397-8200
E-mail: jamie.hohman@garmin.com

Hossein Saiedian, Ph.D., currently is a professor of software engineering at the University of Kansas. His primary area of research is software engineering and, in particular, models for quality software development both technical and managerial. Saiedian has published more than 100 articles on a variety of topics on software engineering and computer science. His past research has been supported by the National Science Foundation as well as regional organizations. Saiedian is a senior member of the Institute for Electrical and Electronic Engineers. Saiedian holds a doctorate degree in computing and information sciences from Kansas State University.

University of Kansas
Department of EECS
Lawrence, KS 66045
Phone: (913) 864-8812
Fax: (913) 897-8682
E-mail: saiedian@ku.edu

Heroes: Carrying a Double-Edged Sword

Paul Kimmerly
U.S. Marine Corps Technology Services Organization

Every organization has key performers that it depends on for its success. Organizations often cast them as heroes that ride in to save the day. These heroes play an important role in getting their organizations through difficult situations, getting products out the door, and keeping customers happy. However, reliance on heroes can create problems just as big as the ones the heroes help resolve. Organizations must recognize the double-edged sword that heroes bring with them. There are ways an organization can leverage the good qualities that heroes bring and minimize the negative ones.

Historian Arthur Schlesinger spoke of the balance between the need for heroes and the problems with hero worship when he said the following:

> To say that there is a case for heroes is not to say that there is a case for hero worship. The surrender of decision, the unquestioning submission to leadership, the prostration of the average man before the Great Man – these are the diseases of heroism, and they are fatal to human dignity. History amply shows that it is possible to have heroes without turning them into gods. And history shows, too, that when a society, in flight from hero worship, decides to do without great men at all, it gets into troubles of its own. [1]

This article discusses how organizations can use heroes to support process improvement efforts and how processes can help alleviate some of the problems created by dependence on heroes.

One of the basic ideas in the Capability Maturity Model® Integration (CMMI®) is that organizations move away from an atmosphere of ad-hoc or even chaotic processes towards an atmosphere where structured processes are in place and everyone looks for ways to continually improve.

An organization operating at an initial level or Level 1 typically has few processes in place. The success of the organization depends on the individual effort of key people or heroes. The Software CMM® states the following:

> Success in a Level 1 organization depends on the competence and heroics of the people in the organization and cannot be repeated unless the same competent individuals are assigned to the next project. [2]

This kind of atmosphere puts a lot of stress on both the heroes and the organization. Low maturity organizations are very dependent on the talent of their heroes. As a result, they are poorly equipped to deal with talent shortages. The heroes are asked to do more and more while others in the organization are asked to do less or given minor assignments. This sets the organization up for potential failure if highly skilled individuals leave. By not establishing processes and involving more members of the organization, it limits its ability to improve performance.

This is especially true as software projects become more complex. In "The Art of War," Sun Tzu said, "In ancient times, those known as heroes prevailed when it was easy to prevail" [3]. The same can be said of software projects. It was easy for heroes to rule the day when software projects were relatively simple and customers were not as demanding. Now that customers have become aware of the capabilities of software systems, they are more demanding. The requested products are more complex. As projects become more complex, the reliance on a small group of heroes to develop the software brings more risk. To alleviate that risk, more people need to be able to contribute to the development effort.

In 1968, Alan J. Perlis told the NATO Science Committee:

> We kid ourselves if we believe that software systems can only be designed and built by a small number of people. If we adopt that view this subject will remain precisely as it is today, and will ultimately die. [4]

The development of software systems has changed a lot since 1968; however, some organizations continue to depend on a small group of heroes in their development efforts. If organizations continue to view software development as an art form practiced by a few individuals, Naur's prediction of doom may come true in terms of unhappy customers and lost business.

The problem with heroes does not rest only with the heroes. Managers may encourage unhelpful behavior by relying too much on their heroes. It is easy for managers to rely too much on heroes to complete complex projects. However, if those projects are left solely in the hands of the heroes, the organization misses an opportunity to build a stronger workforce. The reliance on heroes can make an organization weaker in the long run. By handing complex projects to a hero, an organization can create a single point of failure if that hero is unavailable or if the hero leaves the organization. The days when a small group of experts could be successful by controlling thousands of lines of spaghetti code are long past. Managers create their own problems if they establish such a situation.

The best example of the hero syndrome was told to me by an individual who worked for the Air Force. One of the main systems used by the base where he worked was written, maintained, and completely understood by one individual. Unfortunately, that individual was in a bad car accident, hurting his back, and would be out of the office for a few months while he recovered. No one else knew what to do with the system he maintained. This was before the days of remote connectivity, so the organization had to set up a special terminal and arrange for him to be transported in on a gurney for several weeks so he could show what he did to someone else. It doesn't have to be a major car accident – it could be job turnover or a winning lottery ticket. By placing too much control into the hands of too few people, an organization sets itself up for a potential crisis.

Heroes often hoard information and create a number of potential problems. They can be an impediment to completing work because development activities come

to a halt if they are not available. If they hold too much knowledge, others are unable to complete work unless they get the information they need from the heroes. In extreme cases, an organization can find itself held hostage by its dependence on its heroes. In one situation, a hero was given control over an organization's configuration management system. The hero would change other people's code without their knowledge. The original developer was unable to fix any defects that were identified because the final code did not match what was originally written. The hero had to come in and save the day. Of course, the hero was saving the organization from problems of his own creation. Management rewarded that behavior, which only exacerbated the problem.

To avoid this situation, managers need to coordinate the creative work of their developers so they support rather than interfere with each other. The knowledge and understanding of the system and the processes used to develop it must be spread out across a larger number of people to enable an organization to better deal with complex situations. Heroes can use complexity to their advantage. By making a system so complex that only the heroes understand it, an organization can find itself at risk. The pieces of such complex systems need to be compartmentalized for easier understanding and construction. In such a situation, an organization can use its heroes' expertise and concentrate their attention on the architecture of the system. Getting the heroes focused on the way the system is constructed will enable that organization to leverage its knowledge to more easily maintain pieces for the rest of the development staff.

In his book, "Managing Technical People," Watts Humphrey gives the example of jazz musicians to illustrate the importance of people working together [5]. Individual musicians study at a conservatory to hone their skills through rigorous training. They learn the discipline of playing instruments until a flawless performance is routine. It is only when those musicians come together that beautiful music is created. Jazz musicians have mastered their instruments and techniques to the point that when they play together, they can improvise on a theme to create something truly special. In terms of software development, the discipline of process takes care of the mundane performance issues by establishing routine. With those issues addressed, the process enables more creative work.

This is not to say that heroes and key performers must go away. They exist at every maturity level in the CMMI. The role that heroes play in an organization changes as the organization becomes more mature in process terms. At the lower maturity levels, heroes drive the completion of the work either through their own efforts or by serving as an example for others. Most heroes are successful for a reason. While some are undisciplined, many have developed intuitive processes that they routinely follow. An organization can use their informal processes followed by their heroes as the basis for formal processes that will guide the work of others.

"Managers need to stop rushing from fire to fire and take control of their projects by sticking to the established processes. This means controlling the heroes' natural instinct to jump in and save the day."

In my own organization, estimating was a constant problem. The heroes on one project were called upon to estimate all of the change requests. It was discovered that they all followed a similar intuitive process in creating estimates. The heroes were brought together and their intuitive processes were committed to paper. Values were assigned to the variables they considered in their mental process to create an estimating formula. The heroes bought into the process because they had some structure to fall back on when estimating that matched their intuition. With the formulas in place, management could now task others with deriving estimates, and the heroes had more time to concentrate on development work. By documenting the intuitive processes used by the heroes to be successful, an organization can free up creative time for their heroes because they will not have to answer as many basic questions and others will be able to take some of the more routine work. At the higher maturity levels, heroes are free to become their organization's innovators.

When processes are improvised by the heroes to react to a crisis, it's difficult to maintain process discipline. Processes are often thrown out to deal with the issues of the moment and success is dependent upon the heroes. In this situation, managers have a difficult time understanding the status of their projects because they are reacting to today's crisis. In low maturity organizations, managers can unknowingly abdicate their responsibilities to their heroes in order to get things done. Managers need to stop rushing from fire to fire and take control of their projects by sticking to the established processes. This means controlling the heroes' natural instinct to jump in and save the day. There are times when heroes need to save the day, but those times should not be the norm.

Successfully implemented processes should reflect how the work is being done. The heroes play a role in documenting the processes that will be followed. As mentioned earlier, those processes may come from the successful approach already followed by the heroes. Another way to involve the heroes in the process definition efforts is to cast them in the role of devil's advocate. Since many of the heroes have an understanding of better ways to do things, they can be used to find flaws in the processes being developed. Some heroes may resist change, but using them in the devil's advocate role can engage them in the process improvements. As processes are documented, management needs to ensure that the processes are followed. By establishing and enforcing the use of processes, an organization can reduce some of its dependency on heroes by elevating the performance of the rest of the staff. The heroes can then turn their attention away from fighting fires since fewer fires will exist.

Heroes like challenges. By giving the routine work to others in the organization, managers can channel heroes to more challenging work. With established processes in place, the heroes can be moved to work on other critical projects and the process will survive. The project will not collapse because the single point of expertise is gone. Knowledge is shared throughout the project and the people remaining on the project continue on without loss of quality or understanding. The organization must be sure that the heroes are not left alone to work on the next complex project or the problem situation can happen again.

The establishment of processes allows people in an organization to develop their potential more quickly. Most importantly,

new people are able to become more productive more quickly because the infrastructure is in place to support them. The status of the project is better understood because the process and the infrastructure are in place. Status does not depend upon the knowledge of the heroes. If change is brought into the organization, there is a better chance that it will be successful because the process discipline is understood. Change and the possible improvement it brings are not dependent on whether or not the change matches the heroes' preferences.

Once again, Sun Tzu says it well when he points out the following:

> Good warriors seek effectiveness in battle from the force of momentum, not from individual people. … Therefore, when people are skillfully led into battle, the momentum is like that of round rocks rolling down a high mountain – this is force. [3]

Heroes can be brought into process improvement efforts and used to establish the processes and help the other developers in the organization. The key in using heroes for the good of the organization is management's willingness to change the way they view heroes. Managers need to leverage the good qualities that heroes bring and minimize the hero worship that can create dependence on their worst tendencies. While individual people are important, it is the momentum built by preparing them for whatever task is at hand and giving them the discipline and tools they need to be successful. Then, the leadership can leverage the discipline to build momentum and generate the force Sun Tzu mentions.◆

References

1. Schlesinger, Arthur M. "The Decline of Heroes." <www.quoteland.com>.
2. Paulk, M., et al. The Capability Maturity Model: Guideline for Improving the Software Process. Boston: Addison-Wesley, 1995.
3. Tzu, Sun. The Art of War. Trans. Thomas Cleary. Boston: Shambhala, 1988.
4. Naur, P., and B. Randell. "Software Engineering: Report on a Conference Sponsored by the NATO Science Committee." Garmisch, Germany: 7-11 Oct. 1968.
5. Humphrey, Watts. Managing Technical People. Boston: Addison-Wesley, 1997.

About the Author

Paul Kimmerly has 20 years experience in software development for the different incarnations of the United States Marine Corps Technology Services Organization in Kansas City (TSO-KC/KGB), Mo. A member of the Software Engineering Process Group (SEPG) since 1993, he has served as the group's lead for the past 12 years. Kimmerly is an authorized Standard CMMI Assessment Method for Process Improvement Lead Appraiser. He presented at the 1997 and 2000 Software Engineering Symposiums and the 2004 National SEPG conference, and has contributed several articles on process improvement to CROSSTALK.

TSO-KC/KGB
1500 E 95th ST
Kansas City, MO 64197
Phone: (816) 926-5364
DSN: 465-5364
Fax: (816) 926-6969
E-mail: paul.j.kimmerly@dfas.mil

A Process Improvement Commentary

Dr. Linda Ibrahim
Federal Aviation Administration

This commentary provides some observations regarding process improvement (PI) throughout the past 20 years, offering views related to process models and standards, organizational change, and the PI practitioner.

During the past 20 years, I have worked as a PI practitioner, educator, trainer, consultant, manager, community leader, standards integrator, model builder, and assessor in industry, academia, applied research, and various governments in the United States, Europe, and the Middle East. None of the following comments should be attributed to any particular organization or group I have worked with or for, but represent my synergized views derived from these experiences.

That said, basically the past 20 years have seen the launching of the entire PI field. I have chosen to focus on just a few major themes, although there are several others that could be explored. However, I will restrict myself to some observations in the following three areas:

1. Process models and standards.
2. Organizational change.
3. The PI practitioner.

Process Models and Standards

The Beginning

Let us start with a look at what can be considered the major catalyst for PI: the inception and development of process models, standards, and frameworks. Beginning in the late 1980s, several now-influential process models and standards were born.

For example, we have witnessed the launching of capability maturity models, starting with early work on the Capability Maturity Model® (CMM®) for Software (SW-CMM®) in the late '80s; the International Organization for Standardization (ISO) 9000 Quality Management Systems (1987); the Information Technology (IT) Infrastructure Library (ITIL, late '80s); the Malcolm Baldrige National Quality Award (1987); the ISO/International Electrotechnical Commission (ISO/IEC) 15504 Process Assessment (Software Process Improvement and Capability dEtermination [SPICE], early 1990s); the Internal Control Integrated Framework (1992); the ISO/IEC 12207 – Software Life Cycle Processes (1995); and the Control Objectives for Information and related Technology (1996).

Internationally, the ISO and IEC created a joint technical committee in 1988 to strive for standardization in the field of IT. This committee now has several subcommittees and working groups.

Process standards and frameworks came into existence.

"PI is still fragile, and not yet fully ingrained in many organizational cultures. Strong enterprise leadership and vision remain critical. We continue to need strategic-minded executives to lead and support the path to improved performance ..."

Proliferation

The past 20 years have seen not only the inception but the expansion and evolution of many standards, models, and frameworks in a host of fields. For example, starting with the SW-CMM, the movement to codify process knowledge in the form of a capability or maturity model has exploded to the point where there may be hundreds of such maturity frameworks in existence.

Many other models and standards have been developed through national and international standards bodies, industry, universities, research organizations, and diverse professional communities and societies. We now have a plethora of models, standards, appraisal/assessment methods, and PI approaches that we did not have 20 years ago.

It is important to note that some models and standards that deal with a particular discipline may be mutually supportive and provide guidance at different levels of granularity. However, many are process standards that are at the same or similar level of abstraction with frequent content overlaps but different structures and terminology.

Confusion

What has transpired as a result of all these models and standards? As a student of these endeavors, I have found that each offers guidance useful from the perspective of the model developer, and that these individual efforts have resulted in an immense collection of useful, important, practical guidance. I encourage our professional colleagues to continue to codify good/best practices.

But I am also saddened by the so-called *model wars* that seem to arise as a result: the politics, the competition, the confusion, and the expense that multiple isolated standards can cause.

This is not necessary. We need to rectify these problems for the benefit of our PI customers and the PI community. Such an abundance of disconnected models and standards might impede the adoption of any improvement efforts at all. We in the model-building community need to take heed.

There are several things we have been doing and can do regarding this situation, including model integration, harmonization, and providing guidance for model builders.

Integration and Harmonization

The model proliferation problem was recognized by the late '90s, and efforts were launched to integrate various models. Initial model integration efforts resulted with the release of the Federal Aviation Administration's (FAA) initial integrated Capability Maturity Model (iCMM) in 1997 (integrating the software acquisition, software, and systems engineering CMMs), and with the CMM Integration℠ (CMMI®) initial release in 2000 (integrat-

℠ CMM Integration is a service mark of Carnegie Mellon University.
® SW-CMM is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

ing the software, systems engineering, and integrated product development CMMs).

Both of these model integration initiatives continued to evolve. The iCMM approach has been to extend beyond CMMs by integrating several ISO standards, the Baldrige award, and several safety and security standards into a single integrated model. Meanwhile, the CMMI initiative chose to provide different models for different disciplines resulting in CMMI for development, CMMI for acquisition, and CMMI for services under construction. These multiple CMMI models, or constellations, share common material and a common structure.

Examples of some other integration efforts include the Trillium model, an early effort to embellish the SW-CMM with other standards for telecommunication software product development; the Integrated System Framework, especially intended to reduce costs of appraisals versus multiple standards/models; the Enterprise IT Capability Model, a research project integrating several models; and several initiatives, mostly software or IT specific, within the ISO/IEC 15504 SPICE community.

We also have instances of model builders collaborating that we might mirror. For example, iCMM and CMMI stakeholders collaborated in developing safety and security extensions intended for use with both models. Safety and security experts also collaborated to integrate and harmonize several safety and security standards in this effort. The ISO systems and software lifecycle standards builders are seeking to harmonize their efforts.

Guidance for Model Builders

We need to encourage the codification of good/best practices in many disciplines and domains but we also need to provide guidance for these initiatives. I suggest model builders follow the international requirements for process models as set forth in ISO/IEC 15504 (SPICE). I further invite the experts developing these models to join forces with the Enterprise SPICE initiative (in the following section) so that their discipline-specific efforts can be integrated into Enterprise SPICE and, thus, be made available for use across enterprises internationally.

What Is Next?

In the global arena, the multiple standards problem has been recognized as needing international attention. This resulted in launching the Enterprise SPICE initiative to develop an integrated standards-based model as part of the ISO/IEC 15504 product suite. Enterprise SPICE will provide a single, standards-based model addressing major activities performed across a typical enterprise. It will not be sector-specific and it can be used selectively according to the business objectives of the enterprise.

The Enterprise SPICE model will include all disciplines and source models already integrated into the iCMM and will consider other integrated standards and frameworks that have been developed in Canada, Brazil, and across the global community for the initial baseline model.

Initial decisions for scoping beyond the previous disciplines have been made by the Enterprise SPICE Advisory Board based on analysis of stakeholder inputs. The initial Enterprise SPICE release will also address, at a minimum, service management, human resource management, financial/investment management, knowledge management, environmental standards, and other potential disciplines still under discussion.

"Social and cultural change is needed to bring about the full benefits that standards and models offer to their customers."

Organizational Change

Now let's consider the customer: the organizations and enterprises that the models and standards are developed for to help them be more successful. What has been happening over the past 20 years?

Mixed Adoption

As the standards, models, and frameworks mature, their acceptance continues to rise. Organizations are, to varying degrees, adopting standards, implementing good practices, getting certifications, being audited and appraised, addressing compliance issues, addressing requirements for levels and certifications, improving performance, and measuring the results of improvement activities.

PI is rarely a smooth journey, however. Tides continue to rise and fall, and even though great gains are made and improvements are believed to be institutionalized, management changes and reorganizations may sweep it all away or, worse yet, try to tear it all down. Change is difficult. Managers may prefer the status quo that led to their success, workers may fear layoffs, and rewards may be based on putting out fires rather than preventing them in the first place. It may be difficult to measure the value of improvements made.

PI is still fragile and not yet fully ingrained in many organizational cultures. Strong enterprise leadership and vision remain critical. We continue to need strategic-minded executives to lead and support the path to improved performance; we need business motivation and strong commitment for successful implementation; and we need clear statements as to why an organization is pursuing PI and communication to all regarding their roles and responsibilities in striving for that vision.

Business Objectives

What is your business? What are you trying to achieve? What problems do you need solved? It is important to be clear, as answers to these questions will guide the use of appropriate process standards.

For example, an acquisition organization may seek objectives such as getting value for their money, ensuring alignment of acquired products and services with business/mission need, providing high-quality requirements for acquisitions, minimizing acquisition risk, choosing competent suppliers, and ensuring appropriate oversight over contracts and agreements. Several models and standards have been developed specifically to support acquirers.

Product and service suppliers may seek profit, more business, to remain competitive, to meet client requirements and service level agreements, to enhance customer satisfaction, to improve quality, predictability and productivity, reduce costs, and reduce time-to-market. Again, many models and standards are available to support suppliers.

A public or private sector *enterprise* will typically seek all of these and, additionally, look at business objectives from several broader perspectives such as operational (e.g., reduce time-to-market, increase reliability and productivity); financial (e.g., cost control, better return on assets, meet spend targets); strategic (e.g., customer satisfaction, innovation, improve professional competency); and external (e.g., improve market position, seek recognition in selected areas, address public responsibility).

These objectives and viewpoints need to be tied to a standards-based improve-

ment effort that can help meet these needs.

But the model wars discussed previously persist within our user organizations as well. Multiple compliance issues lead to excessive expense for organizations, adoption of multiple stand-alone models in isolation that can be suboptimal and confusing, and a check-the-box mentality that degrades the value inherent in the use of best practices. Level-chasing sometimes overshadows the business focus that is essential for successful change.

Think Enterprise

From my experience and perspective, most enterprises, large or small, seek excellence and performance results across the enterprise. An enterprise typically will both acquire and supply products and services. You may think that government is just about service and business is just about business but, in fact, government is seeking to operate as a business, and businesses typically provide a service. An enterprise has broad objectives and is likely to engage in many cross-organizational functions and disciplines. It can use best practice guidance regarding them all and is likely to need to address compliance with multiple standards.

Focusing on overall performance improvement remains difficult, however. It is unusual to find a strategic process vision or to see incentives provided for embracing improvement across an enterprise. Even if enterprise-wide improvement is encouraged in a general way, middle managers and staff will work according to their personal directives, which are likely to be more stove-piped in nature. Collaboration might even be penalized.

Social and cultural change is needed to bring about the full benefits that standards and models offer to their customers.

Recommendations

- Use process standards and best practices. They have been shown to be effective.
- Assure alignment with business objectives – this is a critical success factor. Consider what your enterprise is about.
- Use an integrated approach. This will save you time and money; most organizations need to follow three or more standards.
- Improve for real business value not just for certification or to check the box. Does your organization really need ISO 9001 certification? Do your practitioners really need ITIL certification?
- Think *enterprise-wide* to help all aspects of your business.

The PI Practitioner

We now have many process models and many organizations trying to sort them all out and use them to improve the business. So now, over the past 20 years, a new profession has also sprung up: the PI practitioner. Those in this profession include process group leaders, change agents, appraisers, PI trainers, process champions, process engineers, quality managers, process action teams, process owners, facilitators, and change advocates – lots of people trying to help their business by improving its process.

Organizations need help in prioritizing where and how to use standards and best practices, and this is the job of the PI practitioner.

PI practitioners have a dual role: to develop and improve the PI process we own and to use that process to help others improve the processes they own.

PI Knowledge

Through the years, PI practitioners have built up a broad body of knowledge regarding PI. For example, here are some known factors that are critical for successful PI:

1. **Support, commitment, and involvement**
   - Visible support and sustained commitment from senior management.
   - Middle management support and commitment.
   - Grass-roots support and involvement; technical staff involvement.
2. **Showing measurable, observable results**
   - Observable results backed with data to sustain interest and motivation.
   - PI measured, results made visible.
3. **PI management**
   - Effort must be planned and managed.
   - Senior management actively monitors progress.
   - Adequate staff time and resources dedicated.
   - Clear assignment of responsibility.
   - Process group staffed by highly respected people.
   - Risks recognized and mitigated as necessary.
4. **Goals and alignment**
   - Clearly stated, communicated, well understood, appropriate PI goals aligned with the business.
   - Shared values and goals, improvement in everyone's performance plan.
   - Sustained focus and follow through; no constant shifting of priorities.
5. **Knowledge**
   - Having ability, skills, knowledge.
   - Sufficient education about process and PI.
   - For managers, learn enough to manage it and to have confidence in methods used.
6. **Culture**
   - Open communication, teamwork, and mutual trust.
   - Respect for the individual and investment in people.
   - Quality orientation, customer focus, and continuous learning.
   - *Not* a belief that PI gets in the way of real work, and *not* cynicism from previous unsuccessful PI efforts.
   - Culture needs to enable and motivate change.

We know these things, yet they are still difficult to implement and achieve, and many PI initiatives fail. Why? Often it can be traced back to one or more of the previous factors. PI is always a challenge, and this has not changed at all.

Infrastructure

We cannot do it alone. We need a support infrastructure. In this regard, I have been influenced by my experiences abroad. When you work abroad, you are successful when you transition what you know to the nationals, so when you leave they can carry on. My philosophy, in general, is to continue to learn, transfer information, and build an infrastructure of experts that can do the work. In other words, do not be a hero, be a catalyst and a builder, transferring knowledge with the hope of continued improvement.

We need to continue to build our PI infrastructure. We need to infiltrate the organization with PI champions, build in-house skills and a cadre of trainers, process action team leaders and appraisers so the organization is smart, informed, and capable of leading their own improvement efforts.

Challenges

One practitioner challenge that continues to arise is the *flavor-of-the-month* syndrome – the latest fad, newest standards and models, and newest approaches. Do practitioners need to be chameleons and keep changing colors to be successful? I think not. We need to be consistent in our messages: Use best practices, use standards and models, and bring them together to help the business. There will always be new practices that emerge based on

proven use. Rarely, however, do new best practices conflict with previous best practices – it is just an evolution as disciplines mature. There are certain fundamental concepts and principles that do not change, and we need to foster that continuity.

Some Other Advice

Just as we have encouraged organizations to focus on their business objectives, PI practitioners need to do the same by working with the business and treating PI as a strategic initiative when enterprise investment decisions are made. We need to do the following:

- Keep PI aligned with enterprise processes such as investment management and enterprise architecture.
- Consider PI as a strategic service for the enterprise. Know the customer, prepare the business case, work for PI to be in the enterprise portfolio, negotiate service-level agreements, meet PI service levels, and treat processes and PI services as enterprise assets.
- Ensure best practices are addressed in to-be processes prior to investment decisions.
- Minimize investment risks by reviewing internal and external programs for the capability to carry out relevant processes and address weaknesses with corrective action.

More generally, it is important that PI practitioners remain strong. This is not an easy profession, nor necessarily a very popular one, but it can be very rewarding. Continue to motivate and encourage people to take pride in their work and to heed guidance that might help them do their jobs better. Encouraging managers to recognize improvement is all about helping them achieve their objectives and not (as we always hear) getting in the way of the *real work*. Keep it simple. Talk business, not process.

For me, one of the greatest gratifications from PI work has been the reported differences that improved processes have made in people's work lives. Morale has improved, people feel valued, teamwork is enhanced, and people feel good about their jobs when they work together performing sound processes.

Conclusion

Basically, our whole standards-based PI profession has come about over the past 20 years. There has been much advancement in terms of codifying best practices, as organizations have been improving their performance, and practitioners have been developing, implementing, and improving the PI process. There are PI conferences, journals, books, and networks.

I am an integrator, synergizer, and collaborator and my experience has consistently reinforced the value of this approach, which I have put forth in these conclusions.

Model builders and standards developers need to work together. We need to provide sound, robust products that are easy to use and understand for a business. Organizations need to embrace and support the use of standards-based best practices to meet their business needs. Practitioners need to remain persistent and continue to build and use the body of practical PI knowledge.

Model developers, enterprise customers, and practitioners must work together. This should not be difficult since we are all working for the same goals and objectives. We are a team. And as a team we have now formed, done a little storming, and also progressed towards norming. I am convinced the PI community will be performing at an even higher level over the next 20 years.◆

About the Author

Linda Ibrahim, Ph.D., is the International Project Leader for Enterprise SPICE (ISO/IEC 15504) and a founding member of the SPICE Academy. She is also the chief engineer for PI at the FAA and is lead author of the FAA-iCMM, its safety and security extensions, and appraisal method. Ibrahim co-chairs the Enterprise Process Improvement Community of Practice and has served for eight years on the CMMI Steering Group. She has worked in software engineering for several decades in industry, government, and academia in the U.S., Europe, and Middle East. Ibrahim has a bachelor's degree in mathematics, a master's degree in information science, and a doctorate in electrical engineering.

FAA
800 Independence AVE SW
Washington, D.C. 20591
Phone: (202) 267-7443
Fax: (202) 267-9752
E-mail: linda.ibrahim@faa.gov

Web Sites

Agile Manifesto

www.agilemanifesto.org

On February 11-13, 2001, at The Lodge at Snowbird ski resort in the Wasatch Mountains of Utah, 17 people met to talk, ski, relax, and try to find common ground. What emerged was the Agile Software Development Manifesto. Representatives from eXtreme Programming, SCRUM, Dynamic Systems Development Method, Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming, and others sympathetic to the need for an alternative to documentation driven, heavyweight software development processes convened. Currently, a larger gathering of organizational anarchists would be hard to assemble. The emergence of the Manifesto for Agile Software Development symbolizes the participants' intents.

Software Program Managers Network (SPMN)

www.spmn.com

The mission of the SPMN is to identify proven industry and government software best practices and convey them to managers of large-scale software-intensive acquisition programs. Applying extensive *in the trenches* experience, the SPMN enables program managers to achieve project success and deliver quality systems on schedule and on budget. To date, more than 250 Department of Defense programs have benefited directly from SPMN expert consulting.

Java.net

www.java.net

Java.net is the realization of a vision of a diverse group of engineers, researchers, technologists, and evangelists at Sun Microsystems, Inc. to provide a common area for interesting conversations and innovative development projects related to Java technology. The community continues to grow with industry associations, software vendors, universities, and individual developers and hobbyists joining every day. As they meet, share ideas, and use the site's collaboration tools, the communities they form will uncover synergies and create new solutions that render Java technology even more valuable.

Sticky Minds

www.stickyminds.com

StickyMinds.com, a comprehensive online resource for helping produce better software, offers an unrivaled scope of original articles from industry experts, technical papers, industry news, a searchable tools and books guide, discussion forums, and more. StickyMinds.com is the online companion to *Better Software* magazine. StickyMinds.com is the Web's first and most popular interactive community exclusively engaged in improving software quality throughout the software development life cycle. Membership is free.

The Association for Enterprise Integration

www.afei.org

The Association for Enterprise Integration (AFEI) is the leading industry group providing a framework for collaboration between government and industry. The DoD CIO has turned to AFEI to be its conduit for policy and strategy input from industry through jointly chartered working groups. Scheduled events, the resource library, and news can all be accessed without membership on the Web site.

Ten Commandments of COTS

https://acc.dau.mil/CommunityBrowser.aspx?id=24403

Interest in commercial off-the-shelf products requires examination both in terms of its causes and effects, and in terms of its benefits and liabilities. The Defense Acquisition University offers some observations and voices some specific concerns and criticisms. They stress that their observations are essentially cautionary, not condemnatory: Huge growth in software costs will continue, not abate, and appropriate use of commercially available products is one of the remedies that might help to acquire needed capabilities in a cost-effective manner. Where use of an existing component is both possible and feasible, it is no longer acceptable for the government to specify, build, and maintain a comparable product.

National Institute of Standards and Technology: Computer Security Resource Center

www.csrc.nist.gov

The Computer Security Division is one of eight divisions within National Institute of Standards and Technology's (NIST) Information Technology Laboratory. The mission of NIST's Computer Security Division is to improve information systems security by: raising awareness of information technology (IT) risks, vulnerabilities and protection requirements, particularly for new and emerging technologies; researching, studying, and advising agencies of IT vulnerabilities and devising techniques for the cost-effective security and privacy of sensitive federal systems; developing standards, metrics, tests and validation programs; and developing guidance to increase secure IT planning, implementation, management and operation.

The Software Technology Support Center Technical Document Resource

www.stsc.hill.af.mil/resources/tech_docs

An invaluable section of our Web site, this is a repository of guidelines, reports, and templates published over the last several years to help you succeed in defense software engineering. The U.S. Air Force's Software Technology Support Center is excited to provide an updated and condensed version of the "Guidelines for Successful Acquisition and Management of Software Intensive Systems." These guidelines can be found by following the links on the above address.

Joint Program Management Handbook

www.dau.mil/pubs/handbook/handbook.asp

The Joint Program Management Handbook provides a quick guide to assist experienced acquisition professionals assigned to a joint acquisition program. It offers current policy and advice on joint program issues related to service responsibilities, capabilities and requirements, project manager authority, and funding. The views of experienced joint program managers are highlighted within this guide to give practical advice to the reader. Lessons learned and practical guidelines derived from Joint Working Group deliberations and from Defense Acquisition University faculty, who have joint program management experience, have been included.

One Bagel With My Requirements, Please!

I'm pretty much a creature of habit. Every morning, on the way to work, I stop at my local grocery store and get a large coffee with lots of room for cream. I add two French Vanilla creamers, and then add one artificial sweetener (the yellow – *never* the blue or the pink). Finally, I top it off with some skim milk (OK – I use half-and-half, but the skim milk sounded healthier).

Except on weekends. On Saturdays, I stop at ... well, let's call it Albert's Bagels. For years I would walk in and inevitably order a whole-wheat bagel with low-fat cream cheese, smoked salmon, and extra capers. And – of course – a large coffee (see above). There were multiple servers, each one handling one customer at a time. When I got to the front of the line, the pleasant person on the other side of the counter would take my order and make it to my specifications. If the customer at the front of the line had never in his/her life been to a bagel store (and, therefore, took forever to comprehend the intricacies of *plain* or *whole wheat*), another server would be available to handle my order.

That was until a few years ago. I assume that, in an effort to better control inventory and automate things, they went electronic. Now, when you walk into Albert's, you stand in line number one to order from a single person using a hand-held terminal. They actually seem to have two ordering machines, but they never have two working at the same time. There are generally long lines stretching to the door. When you get to the front, the person enters your order into the computer and never seems to know where the extra capers button is. And – occasionally – he or she runs off to help somebody else with production, leaving nobody to take orders until they return. When my order is finally taken, I then go to line number two to pay. The people making the sandwiches now face the other way with their backs to the customer. No more can I say, "Wait – add some more capers." To add insult to injury, people who don't need bagels, but are only ordering a drink have cut ahead of me into line number two. And, inevitably, the person who cut in front of me orders two double mocha cappuccino frappes with extra sprinkles of cinnamon and light nutmeg. The cashier will stop ringing up customers to make the drink – further frustrating me. Of course there are two cashiers' stations – but one cashier is always making a drink or handling the people who cut into the second line *just for a quick bagel*. I've complained about having customers cut – and get dirty looks from the person who cut and blank looks from the staff.

I guess this makes me a curmudgeon[1]. It irritates me that a process that used to work well (and suited the customer) has been updated to *not* work. The person who makes my order isn't the one who took the order, so my order is frequently wrong. Standing in two lines is frustrating. Having people cut into the second line is equally frustrating.

But I understand. The price of ingredients are rising, and Albert's had to economize. Less staff, more work. Electronic ordering makes inventory much easier. Selling *foo-foo* fancy drinks keeps them competitive, and certainly these drinks have a higher profit margin. Still though, it seems something is wrong when loyal customers who like things the way they used to be feel slighted. And since I have lots of time to talk to others while waiting in line, it appears that I'm not the only disgruntled customer.

Isn't it interesting: losing loyal customers to improve the process of the producer? It would appear that making sure the customer is first (or, at the least making sure that the customer *thinks* he or she is first) would be a primary goal. There must have been ways to upgrade service in a manner that would not torque me off every time I walk in.

Updating obsolete software is a fact of life: What is not updated or rewritten to meet new requirements and changing environments becomes obsolete very quickly. New software *will not* have the exact same form, fit, and function. It *can't*. The new software has to accomplish more, in different ways, with new functionality. Like Albert's Bagels, change has to happen to allow them to stay economically competitive. If you're a curmudgeon like me, and you expect release 3.0 of your software to have the same look and feel of version 2.X – well, it's not feasible. If it did look and feel the same, it would be equally obsolete as soon as it was released.

New software must meet the needs of the customer *and* still provide service in a customer-acceptable manner. If you remember to keep customers in the loop as you juggle new environments, limited resources, and changing requirements, then you are doing the best you can! Customers need to understand there will be changes, but as long as the overall mission can still be accomplished, they need to be willing to accept some changes. You know, my bagels still taste just as good – and that extra five minutes really is not that big of a deal. It beats having Albert's go out of business.

Are you keeping your customers in mind as you upgrade and improve? Are the customers involved in changing and building the new system? Will it be acceptable to them? If not, your customers will feel the way I do when I get unordered bean sprouts.

And happy 20th birthday, CROSSTALK.

—David A. Cook, Ph.D.
Principal Member of the Technical Staff and Curmudgeon
The AEgis Technologies Group, Inc.
dcook@aegistg.com

P.S. One very nice thing about Albert's Bagels: They let you get free refills all morning long and pleasantly let you sit with your laptop and write your BACKTALK column without rushing you!

Note

1. Curmudgeon: A bad-tempered, difficult, cantankerous person.

Can You BACKTALK?

Here is your chance to make your point, even if it is a bit tongue-in-cheek, without your boss censoring your writing. In addition to accepting articles that relate to software engineering for publication in CROSSTALK, we also accept articles for the BACKTALK column. BACKTALK articles should provide a concise, clever, humorous, and insightful perspective on the software engineering profession or industry or a portion of it. Your BACKTALK article should be entertaining and clever or original in concept, design, or delivery. The length should not exceed 750 words.

For a complete author's packet detailing how to submit your BACKTALK article, visit our Web site at <www.stsc.hill.af.mil>.

CrossTalk / 517 SMXS/MXDEA
6022 Fir AVE
BLDG 1238
Hill AFB, UT 84056-5820

PRSRT STD
U.S. POSTAGE PAID
Albuquerque, NM
Permit 737

CrossTalk is co-sponsored by the following organizations:

NAVAIR

Homeland Security